

METAHEURISTIC ALGORITHMS

overview, applications, and modifications



Saman M. Almufti

● DeepScience
Open Access Books

Metaheuristics Algorithms: Overview, Applications, and Modifications

Saman M. Almufti

Information Technology Department, Technical College of
Informatics-Akre, Akre University for Applied Sciences, Duhok, Iraq

Department of Computer Science, College of Science, Knowledge
University, Erbil, Iraq



DeepScience

Published, marketed, and distributed by:

Deep Science Publishing
USA | UK | India | Turkey
Reg. No. MH-33-0523625
www.deepscienceresearch.com
editor@deepscienceresearch.com
WhatsApp: +91 7977171947

ISBN: 978-93-7185-710-9

E-ISBN: 978-93-7185-454-2

<https://doi.org/10.70593/978-93-7185-454-2>

Copyright © Saman M. Almufti

Citation: Almufti, S. M. (2025). *Metaheuristics Algorithms: Overview, Applications, and Modifications*. Deep Science Publishing. <https://doi.org/10.70593/978-93-7185-454-2>

This book is published online under a fully open access program and is licensed under the Creative Commons "Attribution-Non-commercial" (CC BY-NC) license. This open access license allows third parties to copy and redistribute the material in any medium or format, provided that proper attribution is given to the author(s) and the published source. The publishers, authors, and editors are not responsible for errors or omissions, or for any consequences arising from the application of the information presented in this book, and make no warranty, express or implied, regarding the content of this publication. Although the publisher, authors, and editors have made every effort to ensure that the content is not misleading or false, they do not represent or warrant that the information-particularly regarding verification by third parties-has been verified. The publisher is neutral with regard to jurisdictional claims in published maps and institutional affiliations. The authors and publishers have made every effort to contact all copyright holders of the material reproduced in this publication and apologize to anyone we may have been unable to reach. If any copyright material has not been acknowledged, please write to us so we can correct it in a future reprint.

Preface

Metaheuristic algorithms have emerged as essential tools for solving complex optimization problems across disciplines such as engineering, logistics, finance, and healthcare. Their ability to efficiently explore large, nonlinear, and uncertain search spaces makes them highly effective where traditional methods often fail.

This book, *Metaheuristics Algorithms: Overview, Applications, and Modifications*, offers a structured overview of key algorithmic families—including evolutionary, swarm-based, physics-inspired, and human-based approaches—supported by theoretical foundations, classifications, and real-world applications. Emphasis is placed on recent advancements, hybridizations, and performance-enhancing modifications.

Designed for students, researchers, and practitioners, the content balances academic rigor with practical relevance, aiming to guide both implementation and innovation in metaheuristic optimization.

I am grateful to my colleagues and reviewers for their valuable input, and to Reta N. Mussa for the attractive design of the book cover. My appreciation also extends to Deep Science Publishing for enabling its open-access dissemination. I hope this work contributes meaningfully to advancing research in computational intelligence and intelligent optimization.

Saman M. Almufti

Information Technology Department, Akre University for Applied Sciences

Department of Computer Science, Knowledge University

Duhok, Kurdistan-Region, Iraq

ORCID: <https://orcid.org/0000-0002-1843-745X>

Table of Contents

Chapter 1: Metaheuristics Algorithms overview1

Chapter 2: Ant Colony Optimization Algorithm.....29

Chapter 3: Lion Algorithm.....40

Chapter 4: Cuckoo Search Algorithm52

Chapter 5: Grey Wolf Optimizer Algorithm63

Chapter 6: Vibrating Particles System Algorithm:77

Chapter 7: Social Spider optimization Algorithm88

Chapter 8: Cat Swarm Optimization Algorithm108

Chapter 9: Bat Algorithm.....121

Chapter 10: artificial bee colony Algorithm134

Chapter 11: Comparative Analysis of Metaheuristic Algorithms for Solving Travelling Salesman Problems152

Chapter 12: Optimization Problem: Models, Methods, and Benchmark Analysis162

Chapter 1: Metaheuristics Algorithms overview

1. Introduction

Metaheuristic algorithms are optimization algorithms that are used to find the optimal solution for complex problems that cannot be solved using traditional methods. These algorithms are inspired by natural phenomena such as genetics, swarm behaviour, and evolution, and they are used to find the global optimum of a problem by exploring a large search space. Some popular metaheuristic algorithms include genetic algorithms, particle swarm optimization, ant colony optimization, simulated annealing, and tabu search. These algorithms have been widely used in various fields such as engineering, finance, and computer science to solve complex problems (Fister, Yang, et al., 2013a). One of the advantages of metaheuristic algorithms is that they do not require an initial starting point for the optimization problem. This makes them particularly useful for solving complex problems where the initial conditions are not known. Additionally, metaheuristic algorithms can handle large and complex search spaces, which traditional optimization methods cannot. However, one of the main drawbacks of metaheuristic algorithms is that they may not always find the global optimum (Almufti et al., 2019). Due to the random nature of the algorithm, there is no guarantee that the algorithm will find the best solution. Additionally, the algorithms can be computationally expensive, especially when dealing with large search spaces. Overall, metaheuristic algorithms are powerful optimization tools that can be used to solve complex problems. However, they should be used with caution and with a thorough understanding of the problem being solved (Almufti et al., 2023). In general, metaheuristic algorithms have several advantages and disadvantages, and the choice of algorithm depends on the characteristics of the optimization problem and the available computing resources (Rai & Tyagi, 2013). Metaheuristic algorithms can provide an effective tool for solving complex optimization problems, but their limitations should also be considered.

2. Optimization

Generally, Optimization refers to the process of finding out the best obtainable solution to a given problem. Optimization Algorithms figure out to be methods of discovering optimized value of a function (objective/fitness function) (Almufti, 2017). This process of optimization is generally performed under some system imposed binding conditions known as constraints. Optimization problem can have single objective or multiple objectives. And it's possible to have many maxima or minima in a search space, indeed all of them are not the best solutions (Zebari et al., 2020). Finding out any of these local optima may disguise the search method of successfully optimizing the problem as can be seen in Fig. 1-1. Global optimization refers to finding out the global optimum avoiding local optima.

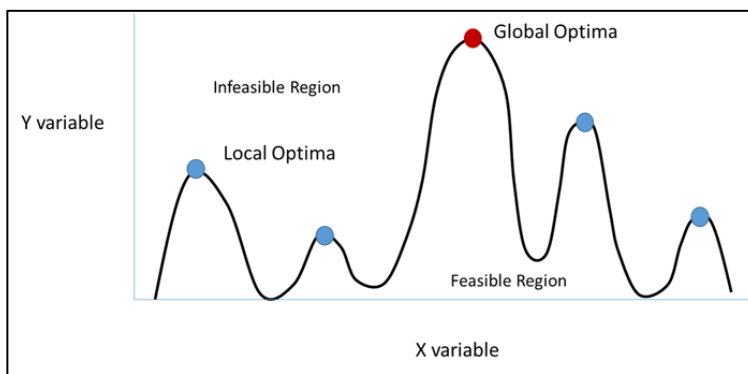


Fig. 1-1: search space local and global optima

One key issue of this process is the immensity of the search space for many real-life problems, in which it is not feasible for all solutions to be checked in a reasonable time.

In Computer Science and Engineering fields many algorithms has been designed for solving optimization problems. Typically Swarm Intelligent algorithms and generally Artificial Intelligent algorithms played important rule in solving these optimization problems, such as solving Travelling Salesman Problem (TSP) which belongs to NP-hard problem by Almufti in (Almufti & Shaban, 2018) by using various swarm intelligent algorithms.

3. Metaheuristic Algorithms

The term "metaheuristics" refers to a "higher level of heuristics" and is a combination of the words "meta" and "heuristic," where "meta" means beyond or higher level and "heuristic" refers to finding or discovering a goal by trial and error. Historically, methods with stochastic mechanisms were frequently referred to as "heuristic algorithms." often, metaheuristic algorithms function as "master strategies that direct and modify other heuristics to produce solutions beyond those that are typically generated in a quest for local optimality". These algorithms modify local search and randomization in a specific

way. In a fair amount of time, excellent solutions to challenging optimization problems can be developed. But in general, there is no guarantee of finding optimal solutions(S. Almufti, 2017).

The term "metaheuristic" refers to a higher-level procedure or heuristic in the fields of computer science, mathematical optimization, and engineering that is used to search for, find, generate, or select a heuristic that may offer a good solution to an optimization problem, particularly for large problems like (NP-hard problem) or in cases of limit, incomplete, or imperfect information. The collection of solutions in a metaheuristic is too big to sample entirely. Metaheuristics can be used for different problems since they may not make many assumptions about the optimization problem being handled(Almufti, 2022a). In contrast to iterative or optimization techniques, metaheuristics do not ensure that the optimum solution for a given class of problems can be identified. Numerous metaheuristics use stochastic optimization in some way, which means that the solution is based on the set of produced random variables(Ihsan et al., 2021). Metaheuristics may frequently identify good solutions in combinatorial optimization with less computing work than optimization algorithms, iterative techniques, or basic heuristics since they search through a vast range of viable alternatives. Because of this, they are effective strategies for solving optimization issues (Sadeeq et al., 2021).

Most literature on metaheuristics is experimental in nature, describing empirical results based on computer experiments with the algorithms. But some formal theoretical results are also available, often on convergence and the possibility of finding the global optimum. Many metaheuristic methods have been published with claims of novelty and practical efficacy. While the field also features high-quality research, many of the publications have been of poor quality; flaws include vagueness, lack of conceptual elaboration, poor experiments, and ignorance of previous literature (Agrawal et al., 2021).

The properties that characterize most metaheuristics:

- Metaheuristics are strategies that guide the search process.
- The goal is to efficiently explore the search space in order to find near-optimal solutions.
- Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- Metaheuristics are not problem-specific, Which mean it can be used for solving various problem.

4. Well-Known Metaheuristic Algorithms

Over years, various metaheuristic algorithms have been developed, each with its own strengths and weaknesses. In this section some of the most well-known metaheuristic algorithms, along with a brief description of how they work are outlined:

- A. Genetic Algorithms (GA): GA is a metaheuristics algorithm inspired by the principles of natural selection and genetics. It starts by randomly generating an initial population of solutions, and then evolves the population by applying operators such as selection, crossover, and mutation. These operators simulate the process of natural selection, where individuals with higher fitness are more likely to reproduce and pass on their traits to the next generation. GA has been widely used in optimization problems that involve finding the best combination of parameters or features(Acan et al., n.d.).
- B. Particle Swarm Optimization (PSO): PSO is a swarm intelligence algorithm that is inspired by the collective behaviour of social organisms, such as birds flocking or fish schooling. It starts by randomly generating a swarm of particles, each representing a potential solution to the optimization problem. Each particle moves in the search space based on its own position and velocity, as well as the best position found by the swarm so far. PSO has been widely used in optimization problems that involve finding the best configuration of weights in neural networks or other machine learning models(Almufti, Yahya Zebari, et al., 2019).
- C. Simulated Annealing (SA): SA is a stochastic optimization algorithm that is inspired by the process of annealing in metallurgy. It starts by randomly generating an initial solution, and then gradually reduces the temperature of the system. As the temperature decreases, the algorithm becomes more likely to accept worse solutions in order to escape local optima. SA has been widely used in optimization problems that involve finding the best configuration of parameters in complex models or simulations(Asaad & Abdulnabi, 2018).
- D. Tabu Search (TS): TS is a metaheuristic algorithm that is inspired by the concept of memory in human decision-making. It starts by randomly generating an initial solution, and then searches the neighbourhood of the solution by applying operators such as swapping or reversing. The algorithm uses a tabu list to remember recently visited solutions and avoids revisiting them. TS has been widely used in optimization problems that involve finding the best sequence of actions or decisions, such as scheduling or routing(Acan & Ünveren, 2015; Ridwan et al., 2020).
- E. Ant Colony Optimization (ACO): ACO is a swarm intelligence algorithm that is inspired by the behaviour of ants in finding the shortest path between their nest and a food source. It starts by randomly generating an initial set of pheromone trails, which represent the quality of the solutions found so far. As ants move

through the search space, they deposit or follow pheromone trails based on the quality of the solutions they find. ACO has been widely used in optimization problems that involve finding the best routes or paths, such as in logistics or transportation(Agrawal et al., 2021).

These are just a few examples of the many metaheuristic algorithms that have been developed over the years. Each algorithm has its own strengths and weaknesses, and the choice of algorithm depends on the characteristics of the optimization problem and the available computing resources(Dubey et al., 2014; Fister, Yang, et al., 2013a; Mishra & Mishra Scholar, 2017).

The following table summarizes some widely used metaheuristic algorithms, their inspirations, and their key features(Dokeroglu et al., 2019; M. Almufti, Boya Marqas, et al., 2019; Yang, 2010):

Table 1-1: popular Metaheuristics Algorithms

Algorithm	Inspiration	Key Features	Common Applications
Genetic Algorithm (GA)	Evolutionary processes (natural selection, crossover, mutation)	Effective for combinatorial and discrete optimization problems; handles noisy functions well; parallelizable.	Scheduling, neural network optimization, portfolio optimization.
Particle Swarm Optimization (PSO)	Social behaviour of birds and fish (swarm intelligence)	Fast convergence for continuous problems; easy implementation; requires fewer parameters.	Function optimization, clustering, robotic path planning.
Simulated Annealing (SA)	Annealing process in metallurgy (cooling of metals)	Robust against local optima; well-suited for single-solution problems with constraints.	Circuit design, job scheduling, layout optimization.
Ant Colony Optimization (ACO)	Foraging behaviour of ants (pheromone-based communication)	Effective for combinatorial and graph-based problems; builds solutions incrementally.	Traveling Salesman Problem (TSP), network routing, logistics optimization.
Cuckoo Search Algorithm (CSA)	Brood parasitism of cuckoo birds	Combines simplicity with strong global search capabilities; excels in multi-modal problems.	Structural design, feature selection, power flow optimization.
Differential Evolution (DE)	Evolutionary processes (mutation, crossover, selection)	Effective for multi-objective and continuous optimization; robust	Multi-objective optimization,

Firefly Algorithm (FA)	recombination, selection) Bioluminescent communication of fireflies	against non-differentiable functions. Attractiveness-based search for multi-modal problems; good for fine-tuning solutions.	engineering design, control systems. Image processing, clustering, energy optimization.
Harmony Search (HS)	Musical improvisation (harmony memory, pitch adjustment)	Simple to implement; excellent for discrete and mixed-variable optimization.	Structural optimization, energy management, scheduling.
Tabu Search (TS)	Memory-based search (prohibiting recently visited solutions)	Avoids cycling through local optima; uses adaptive memory for dynamic solution improvement.	Scheduling, industrial design, resource allocation.
Bee Algorithm (BA)	Foraging behaviour of honeybees	Efficient local search with good balance of exploration and exploitation; handles multiple constraints.	Data clustering, job-shop scheduling, transportation.
Whale Optimization Algorithm (WOA)	Hunting strategy of humpback whales (bubble-net feeding)	Strong performance on continuous problems; good at escaping local optima.	Feature selection, energy optimization, medical imaging.
Grey Wolf Optimizer (GWO)	Leadership hierarchy and hunting behaviour of grey wolves	Minimal parameters; excels in local exploitation.	Power systems optimization, control problems, financial modelling.
Bat Algorithm (BA)	Echolocation behaviour of bats	Flexible adaptation to constraints; efficient on non-linear optimization problems.	Speech recognition, function optimization, biomedical applications.
Cultural Algorithm (CA)	Cultural evolution in societies	Employs knowledge-sharing mechanisms for improved learning and convergence.	Evolutionary robotics, adaptive system modeling.
Memetic Algorithm (MA)	Hybrid of genetic algorithms and local search heuristics	Combines global search of GA with local refinement; high-quality solutions.	Scheduling, machine learning hyperparameter tuning.
Artificial Bee Colony (ABC)	Foraging behaviour of honeybees	Lightweight and robust; well-suited for dynamic optimization.	Data mining, system identification, scheduling.

Crow Search Algorithm (CSA)	Intelligent hiding behaviour of crows	Excellent diversity in search; balances exploitation and exploration effectively.	Image processing, engineering optimization, signal processing.
Flower Pollination Algorithm (FPA)	Pollination behaviour of flowering plants	Simple implementation; efficient for multi-objective problems.	Renewable energy systems, engineering design, image segmentation.
Wolf Pack Algorithm (WPA)	Hunting behaviour of wolves in packs	Strong local refinement; effective coordination of multiple solutions.	Multi-modal optimization, mechanical engineering.
Shuffled Frog Leaping Algorithm (SFLA)	Mimics leaping behaviour of frogs in groups	Combines global search with local improvement; adaptable to constraints.	Wastewater treatment optimization, project scheduling, hydrology modelling.
Bacterial Foraging Optimization (BFO)	Chemotactic behaviour of bacteria	Strong in adaptive optimization under dynamic environments.	Sensor placement, bioinformatics, resource scheduling.
Multi-Verse Optimizer (MVO)	Big Bang theory (cosmological physics)	Balances exploration and exploitation with good global search capabilities.	Structural optimization, energy management, job-shop scheduling.
Ant Lion Optimizer (ALO)	Hunting behaviour of ant lions in sand traps	Well-suited for solving high-dimensional optimization problems.	Feature selection, parameter tuning, design problems.

Table 1 Shows

- **Inspiration:** Highlights the natural or physical process the algorithm models.
- **Key Features:** Discusses the algorithm's strengths, such as adaptability, robustness, or computational efficiency.
- **Common Applications:** Lists areas where each algorithm is frequently applied, showcasing their practical use cases.

5. Metaheuristic Algorithms Classifications

Metaheuristic algorithms can be classified in several ways based on different criteria. Here are some common classifications (S. Almufti, 2017, 2021; Almufti et al., 2019):

- A. Nature-inspired vs. non-nature-inspired: Metaheuristics can be classified based on whether they are inspired by natural processes or not. Nature-inspired metaheuristics include algorithms such as genetic algorithms, particle swarm

optimization, and ant colony optimization. Non-nature-inspired metaheuristics include algorithms such as simulated annealing and tabu search.

- B. Single-solution vs. population-based: Metaheuristics can also be classified based on whether they operate on a single solution or a population of solutions. Single-solution metaheuristics include algorithms such as simulated annealing and hill climbing. Population-based metaheuristics include algorithms such as genetic algorithms, particle swarm optimization, and ant colony optimization.
- C. Deterministic vs. stochastic: Metaheuristics can be classified based on whether they use deterministic or stochastic processes. Deterministic metaheuristics use deterministic processes to generate new solutions. Examples include hill climbing and deterministic annealing. Stochastic metaheuristics use stochastic processes to generate new solutions. Examples include genetic algorithms and simulated annealing.
- D. Trajectory-based vs. population-based: Metaheuristics can also be classified based on whether they focus on finding a single solution or multiple solutions. Trajectory-based metaheuristics focus on finding a single solution and use iterative improvement to refine that solution. Examples include hill climbing and simulated annealing. Population-based metaheuristics focus on finding multiple solutions and use a population of solutions to explore the search space. Examples include genetic algorithms and particle swarm optimization.
- E. Local search-based vs. global search-based: Metaheuristics can be classified based on whether they focus on exploring the local search space or the global search space. Local search-based metaheuristics focus on finding solutions in the immediate vicinity of the current solution. Examples include hill climbing and tabu search. Global search-based metaheuristics focus on exploring the entire search space. Examples include genetic algorithms and ant colony optimization.

Unlike traditional optimization techniques that rely on mathematical models and assumptions, metaheuristics are problem-independent and can adapt to different types of problems with minimal modifications.

The key features of metaheuristic algorithms are (Bhuvaneswari et al., 2014; Fister, Yang, et al., 2013b; Selvaraj & Kumar, n.d.):

- A. Exploration and exploitation: Metaheuristics balance the exploration of the search space to discover new promising solutions and the exploitation of the already discovered solutions to refine them further.
- B. Stochastic search: Metaheuristics use randomization to generate new solutions and avoid being trapped in local optima.
- C. Iterative improvement: Metaheuristics improve the quality of the solutions iteratively by repeatedly applying modifications to the existing solutions.

- D. **Robustness:** Metaheuristics are designed to handle noisy and uncertain problem instances by avoiding the dependence on specific problem characteristics and assumptions.
- E. **Flexibility:** Metaheuristics can be customized and adapted to different problem domains by changing the selection criteria, neighbourhood structures, and search strategies.
- F. **Parallelism:** Metaheuristics can be easily parallelized to speed up the search process and solve large-scale problems efficiently.

The diagram in Fig. 1-2, presents a hierarchical classification of metaheuristic algorithms based on their underlying sources of inspiration. At the top level, it identifies the broad category of metaheuristic algorithms, which is then divided into four primary classes: evolution-based, swarm intelligence-based, physics-based, and human behaviour-related algorithms. Evolution-based algorithms

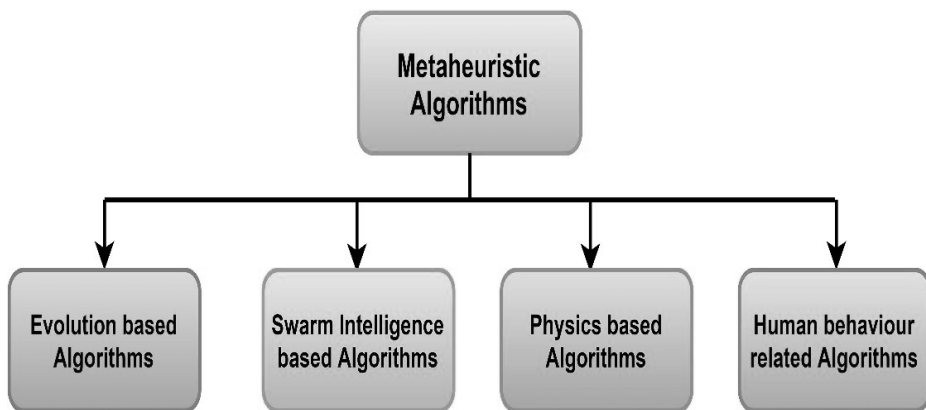


Fig. 1-2: Metaheuristic Algorithm classifications

5.1. Evolutionary based algorithms

Evolutionary algorithms (EAs) are a class of metaheuristic algorithms that are inspired by the process of natural evolution. They are based on the principles of survival of the fittest, reproduction, and mutation. EAs work by creating a population of potential solutions to a problem and then iteratively evolving and improving that population through selection, crossover, and mutation operations (Bäck & Schwefel, 1993; Bartz-Beielstein et al., 2014).

Some common types of evolutionary algorithms include:

- a. **Genetic algorithm (GA):** GA is one of the most widely used EAs. It is based on the principles of natural selection and genetic inheritance. GA

has been applied to a wide range of problems such as optimization, classification, and clustering(Acan et al., n.d.; Luis & Sequera, n.d.).

- b. Evolutionary programming (EP): EP is a stochastic optimization technique that is used to solve a wide range of problems. It is based on the idea of genetic algorithms and is typically used for function optimization(Bäck & Schwefel, 1993; Bartz-Beielstein et al., 2014; Sinha et al., 2003).
- c. Evolution strategies (ES): ES is a family of optimization techniques that are based on the idea of natural evolution. It is used for function optimization and is known for its ability to handle noise in the fitness function(Sun et al., 2009).
- d. Differential evolution (DE): DE is an optimization algorithm that is used to solve problems in continuous search spaces. It is based on the idea of differential mutation and is known for its robustness and efficiency(Sadeeq et al., 2021).

Evolutionary algorithms have been applied to a wide range of problems in fields such as engineering, finance, biology, and computer science. They are particularly useful for solving problems that involve high-dimensional search spaces, nonlinearity, and noisy or imprecise fitness functions(Sharif et al., 2009).

5.2. Physics-based Algorithm

Physics-based metaheuristic algorithms are a class of metaheuristic algorithms that are inspired by physical laws and principles. These algorithms use concepts from physics, such as energy and force, to guide the search for optimal solutions in a problem space(Zhengming Wan & Zhao-Liang Li, 1997).

Some examples of physics-based metaheuristic algorithms are:

- a. Gravitational Search Algorithm (GSA): This algorithm is inspired by the law of gravity and the motion of celestial bodies. In GSA, the solutions are treated as particles that interact with each other through gravitational forces. The algorithm uses these forces to update the positions of the particles in the search space, with the goal of finding the optimal solution(Rashedi et al., 2009).
- b. Electromagnetic Field Optimization (EMO): This algorithm is inspired by the behaviour of electromagnetic fields. In EMO, the solutions are treated as charged particles that interact with each other and with an electromagnetic field. The algorithm uses these interactions to update the positions of the particles, with the goal of finding the optimal solution(Ahmad, 2022).

- c. Quantum-inspired Evolutionary Algorithm (QEA): This algorithm is inspired by the principles of quantum mechanics. In QEA, the solutions are represented as quantum states, and the search process is guided by quantum operators such as rotation and reflection. The algorithm uses these operators to explore the search space and find the optimal solution(Almufti, 2015).
- d. Harmony Search Algorithm (HSA): This algorithm is inspired by the process of musical improvisation. In HSA, the solutions are treated as musical notes that are combined to form a melody. The algorithm uses a process of improvisation and adaptation to generate new melodies, with the goal of finding the optimal solution(Yang, n.d.).

Physics-based metaheuristic algorithms have been successfully applied to a wide range of optimization problems, including engineering design, scheduling, and image processing. They offer a unique approach to optimization that can be particularly effective in problems with complex search spaces and multiple objectives.

5.3. Swarm Based algorithm

Swarm intelligence is a branch of artificial intelligence that is inspired by the behaviour of social animals, such as ants, bees, and birds. It refers to the collective behaviour of decentralized, self- organized systems, where individual agents interact with each other and with their environment to achieve a common goal(S. Almufti, 2017; Sherinov & Ünveren, 2018).

Swarm intelligence algorithms are typically based on the following principles:

- a. Decentralization: There is no centralized control over the system. Instead, each agent follows simple rules that are based on local information and interaction with its neighbours.
- b. Self-organization: The agents interact with each other and with their environment to form a pattern or structure that emerges from the collective behaviour of the system.
- c. Adaptation: The system is capable of adapting to changes in its environment, either through the individual behaviour of the agents or through the emergence of new collective patterns.

Some examples of swarm intelligence algorithms include (Shaban, 2023):

- d. Ant colony optimization (ACO): A metaheuristic algorithm inspired by the behaviour of ants searching for food. It is used to solve optimization problems such as the traveling salesman problem(S. M. Almufti, n.d.-a).

- e. Particle swarm optimization (PSO): A metaheuristic algorithm inspired by the flocking behaviour of birds. It is used to solve optimization problems such as function optimization and feature selection (Kennedy & Eberhart, n.d.).
- f. Bee colony optimization (BCO): A metaheuristic algorithm inspired by the foraging behaviour of honeybees. It is used to solve optimization problems such as function optimization and feature selection.
- g. (Fister, Fister, et al., 2013) algorithm (FA): A metaheuristic algorithm inspired by the flashing behavior of fireflies. It is used to solve optimization problems such as function optimization and clustering.
- h. Artificial bee colony (ABC): A metaheuristic algorithm inspired by the foraging behavior of honeybees. It is used to solve optimization problems such as function optimization and feature selection (S. M. Almufti, Alkurdi, et al., n.d.).

Swarm intelligence algorithms have found applications in many fields, such as engineering, finance, biology, and computer science. They are particularly useful in solving complex optimization problems that are difficult to solve using traditional optimization methods.

5.4. Human-based algorithms

Human-based metaheuristic algorithms are a class of optimization techniques that incorporate human intelligence, knowledge, and experience into the optimization process. Some examples of human-based metaheuristic algorithms are (Dehghani et al., 2022):

- a. Expert-guided Evolutionary Algorithm (EEA): The EEA is a metaheuristic optimization technique that combines evolutionary algorithms with expert knowledge. In this algorithm, the expert knowledge is used to guide the evolution process towards solutions that are likely to be good (Qian et al., 2017).
- b. Human-guided Search (HGS) algorithm: The HGS algorithm is a metaheuristic optimization technique that incorporates human guidance into the search process. In this algorithm, a human expert provides guidance to the optimization process, which helps to direct the search towards promising areas of the solution space (Klau et al., 2010).
- c. Interactive Evolutionary Computation (IEC) algorithm: The IEC algorithm is a metaheuristic optimization technique that involves interaction between the optimization algorithm and a human evaluator. In this algorithm, the optimization algorithm generates candidate solutions, and the human evaluator provides feedback on the quality of

the solutions. The feedback is then used to refine the search process(Marques et al., 2010).

- d. Human-in-the-Loop Optimization (HILO) algorithm: The HILO algorithm is a metaheuristic optimization technique that involves active participation of humans in the optimization process. In this algorithm, humans interact with the optimization algorithm in real-time, providing feedback and guidance to the algorithm as it searches for solutions(Zhang et al., 2017).

These algorithms are examples of how human-based optimization techniques can be effective in solving complex optimization problems. By incorporating human intelligence, knowledge, and experience into the optimization process, these algorithms can often find high-quality solutions more quickly and efficiently than traditional optimization techniques.

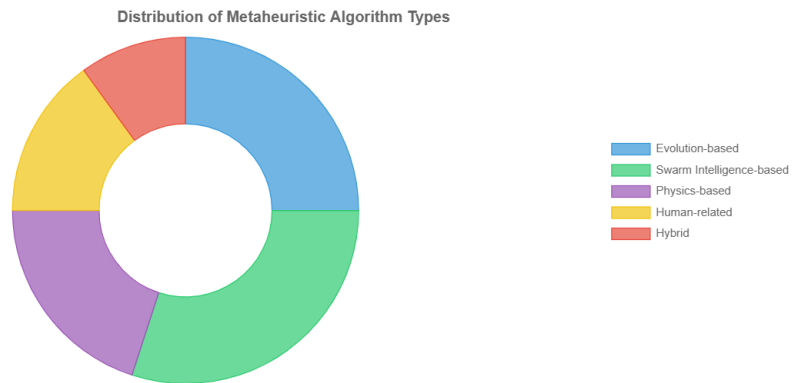


Fig. 1-3:Distribution of metaheuristics algorithm

Fig. 1-3, donut chart illustrates the categorization of metaheuristic algorithms based on their underlying source of inspiration. The chart divides the algorithms into five distinct groups: evolution-based, swarm intelligence-based, physics-based, human-related, and hybrid. Evolution-based algorithms, such as Genetic Algorithms and Differential Evolution, draw inspiration from natural selection and biological evolution. Swarm intelligence-based algorithms, including Particle Swarm Optimization and Ant Colony Optimization, are modelled on the collective behaviour of social organisms. Physics-based algorithms, such as Simulated Annealing and Gravitational Search Algorithm, are grounded in physical phenomena and natural laws. Human-related algorithms mimic cognitive or social behaviour, such as Teaching–Learning-Based Optimization. Finally, hybrid algorithms combine elements from multiple categories to enhance performance and adaptability. The chart provides a visual summary of the algorithmic landscape, emphasizing the diversity and interdisciplinary nature of metaheuristic development.

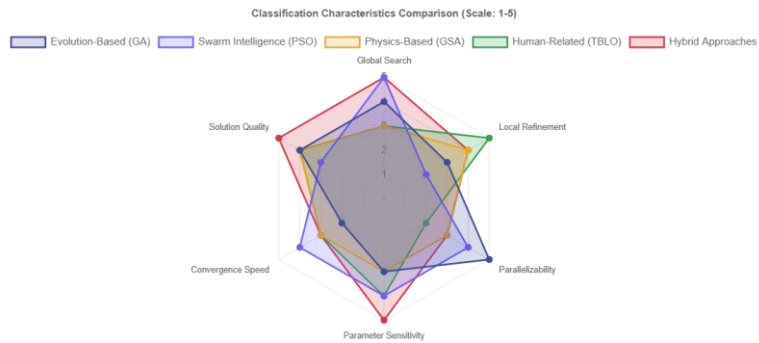


Fig. 1-4:classification characteristics comparison

The radar chart in Fig.1-4 compares classification characteristics of metaheuristic algorithm types across six criteria on a scale of 1 to 5. Hybrid approaches score highest overall, particularly in solution quality, parameter diversity, and global search. Swarm intelligence-based algorithms (e.g., PSO) excel in convergence speed and global search, while human-related algorithms lead in local refinement. Evolution-based and physics-based methods show balanced but generally lower performance across most dimensions, highlighting the strength of hybrids in combining the advantages of multiple strategies.

6.Applications of metaheuristics algorithms

Metaheuristics algorithms have been applied to a wide range of optimization problems in various fields, such as engineering, finance, logistics, and healthcare. In this report, we will discuss some of the notable applications of metaheuristics algorithms(Mohammed Almufti et al., 2022; Rere et al., 2016; Sherinov et al., 2011; Sun et al., 2009).

6.1. Engineering:

Metaheuristics algorithms have been used to solve various engineering problems, such as designing efficient structures, optimizing manufacturing processes, and scheduling production. For example, in the design of structures, genetic algorithms and particle swarm optimization have been used to optimize the shape and material of the structure to reduce weight and improve performance. In manufacturing, simulated annealing has been applied to optimize the cutting parameters of a CNC machine, while ant colony optimization has been used to optimize the layout of a production line(S. M. Almufti et al., 2018; Dhiman & Kumar, 2017).

Generally, Metaheuristics algorithms have been widely used in engineering for solving various optimization problems. Here are some of the notable applications of metaheuristics algorithms in engineering(S. M. Almufti, 2022b):

- a. Design optimization: Metaheuristics algorithms such as genetic algorithms and particle swarm optimization have been used to optimize the design of various engineering structures, such as aircraft wings, bridges, and buildings. These algorithms can optimize the shape and material of the structure to reduce weight, improve performance, and minimize cost.
- b. Manufacturing optimization: Metaheuristics algorithms such as simulated annealing and tabu search have been applied to optimize manufacturing processes. For example, these algorithms can optimize the cutting parameters of a CNC machine, reduce tool wear and machining time, and improve the quality of the manufactured parts.
- c. Supply chain optimization: Metaheuristics algorithms such as genetic algorithms and ant colony optimization have been used to optimize supply chain operations, such as production planning, inventory management, and transportation. These algorithms can optimize the allocation of resources, reduce transportation costs, and improve delivery times.
- d. Energy system optimization: Metaheuristics algorithms such as genetic algorithms and particle swarm optimization have been used to optimize the design and operation of energy systems, such as power grids, renewable energy systems, and energy storage systems. These algorithms can optimize the system parameters, such as the capacity of power plants and the location of renewable energy sources, to maximize efficiency and minimize cost.
- e. Process optimization: Metaheuristics algorithms such as simulated annealing and tabu search have been used to optimize chemical and industrial processes. These algorithms can optimize the process parameters, such as temperature, pressure, and flow rate, to maximize the yield and quality of the products.
- f. Structural optimization: Metaheuristics algorithms such as genetic algorithms and particle swarm optimization have been used to optimize the topology, size, and shape of structures, such as trusses and frames, to minimize weight, reduce material usage, and improve strength.

6.2. Finance:

Metaheuristics algorithms have been applied to financial problems, such as portfolio optimization, risk management, and algorithmic trading. In portfolio optimization, genetic algorithms and particle swarm optimization have been used to optimize the allocation of assets in a portfolio to maximize returns while minimizing risk. In risk management, simulated annealing has been applied to optimize the hedging strategy of a financial institution, while ant colony

optimization has been used to optimize the allocation of insurance policies to customers(Soler-Dominguez et al., 2018).

Generally, Metaheuristics algorithms have been widely used in finance for solving various optimization problems. Here are some of the notable applications of metaheuristics algorithms in finance:

- a. Portfolio optimization: Metaheuristics algorithms such as genetic algorithms and particle swarm optimization have been used to optimize the allocation of assets in a portfolio to maximize returns while minimizing risk. These algorithms can find the optimal portfolio composition by considering multiple factors such as return, volatility, and correlation among assets.
- b. Risk management: Metaheuristics algorithms such as simulated annealing and tabu search have been applied to optimize the hedging strategy of a financial institution. These algorithms can identify the optimal combination of financial instruments to mitigate the risk associated with market fluctuations.
- c. Algorithmic trading: Metaheuristics algorithms such as genetic algorithms and ant colony optimization have been used to develop trading strategies that can generate profits by exploiting market inefficiencies. These algorithms can analyze large amounts of data and identify trading opportunities in real-time.
- d. Credit scoring: Metaheuristics algorithms such as genetic algorithms and particle swarm optimization have been used to develop credit scoring models that can predict the probability of default for a borrower. These algorithms can analyse multiple factors such as credit history, income, and employment status to provide a more accurate assessment of creditworthiness.
- e. Fraud detection: Metaheuristics algorithms such as simulated annealing and tabu search have been used to detect fraudulent activities in financial transactions. These algorithms can identify patterns and anomalies in transaction data to identify potential fraudsters.
- f. Forecasting: Metaheuristics algorithms such as genetic algorithms and ant colony optimization have been used to develop forecasting models for financial time series data. These algorithms can analyze historical data and identify trends and patterns to make predictions about future market conditions.

6.3. Logistics:

Metaheuristics algorithms have been used to solve various logistics problems, such as vehicle routing, inventory management, and facility location. For example, in

vehicle routing, genetic algorithms and tabu search have been used to optimize the routes of delivery trucks to reduce transportation costs and improve delivery times. In inventory management, simulated annealing has been applied to optimize the ordering policies of a retailer to minimize inventory holding costs while ensuring customer satisfaction. In facility location, ant colony optimization has been used to optimize the location of warehouses and distribution centers to minimize transportation costs and improve efficiency(Gogna & Tayal, 2013).

In general, Metaheuristics algorithms have been widely used in logistics for solving various optimization problems. Here are some of the notable applications of metaheuristics algorithms in logistics:

- a. Vehicle routing: Metaheuristics algorithms such as genetic algorithms and tabu search have been used to optimize the routing of delivery vehicles to minimize distance traveled and delivery time. These algorithms can consider multiple factors such as traffic congestion, delivery time windows, and vehicle capacity to find the optimal route.
- b. Facility location: Metaheuristics algorithms such as simulated annealing and ant colony optimization have been applied to optimize the location of warehouses, distribution centers, and production facilities. These algorithms can consider multiple factors such as transportation costs, demand, and labor costs to find the optimal location.
- c. Inventory management: Metaheuristics algorithms such as genetic algorithms and particle swarm optimization have been used to optimize inventory levels to reduce costs while maintaining service levels. These algorithms can consider multiple factors such as demand variability, lead time, and ordering costs to find the optimal inventory level.
- d. Supply chain network design: Metaheuristics algorithms such as simulated annealing and tabu search have been used to optimize the design of supply chain networks to reduce costs and improve efficiency. These algorithms can consider multiple factors such as transportation costs, facility costs, and demand variability to find the optimal network design.
- e. Container loading: Metaheuristics algorithms such as genetic algorithms and particle swarm optimization have been used to optimize the loading of containers to maximize space utilization and minimize transportation costs. These algorithms can consider multiple factors such as container dimensions, cargo characteristics, and loading constraints to find the optimal loading plan.
- f. Warehouse layout optimization: Metaheuristics algorithms such as simulated annealing and ant colony optimization have been used to optimize the layout of a warehouse to improve efficiency and reduce

operational costs. These algorithms can consider multiple factors such as storage capacity, product flow, and material handling costs to find the optimal layout.

6.4. Healthcare:

Metaheuristics algorithms have been applied to healthcare problems, such as patient scheduling, resource allocation, and treatment planning. For example, in patient scheduling, simulated annealing has been applied to optimize the scheduling of patients in a hospital to reduce waiting times and improve resource utilization. In resource allocation, genetic algorithms and particle swarm optimization have been used to optimize the allocation of healthcare resources, such as hospital beds and medical equipment, to maximize the quality of care. In treatment planning, ant colony optimization has been used to optimize the radiation therapy plan for cancer patients to minimize side effects and improve treatment outcomes(Almufti, 2022; Dhiman & Kumar, 2017; Marashdih et al., 2018).

Metaheuristic algorithms have been successfully applied in various healthcare domains, such as medical diagnosis, treatment planning, drug discovery, and healthcare management. Some specific applications of metaheuristic algorithms in healthcare include:

- a. Medical image analysis: Metaheuristic algorithms can be used to analyse medical images and extract useful information for disease diagnosis and treatment planning. For example, genetic algorithms can be used to optimize the segmentation of brain tumors from magnetic resonance imaging (MRI) scans.
- b. Drug discovery: Metaheuristic algorithms can be used to optimize the molecular structure of new drugs and predict their efficacy and safety. For example, particle swarm optimization can be used to optimize the docking of ligands to protein targets in order to discover new drug candidates.
- c. Healthcare scheduling: Metaheuristic algorithms can be used to optimize healthcare scheduling and resource allocation, such as staff scheduling, patient appointment scheduling, and bed allocation. For example, ant colony optimization can be used to optimize nurse scheduling in a hospital.
- d. Disease diagnosis: Metaheuristic algorithms can be used to diagnose diseases based on patient symptoms and medical history. For example, genetic algorithms can be used to diagnose diabetes based on patient data such as age, weight, and blood glucose levels.

- e. Medical equipment design: Metaheuristic algorithms can be used to optimize the design of medical equipment, such as prosthetic limbs and medical implants. For example, simulated annealing can be used to optimize the design of a prosthetic limb for a patient with a missing limb.

Overall, metaheuristic algorithms have the potential to revolutionize the healthcare industry by providing efficient and effective solutions to complex healthcare problems.

6.5. Computer Science:

Metaheuristic algorithms have a wide range of applications in the field of computer science, including optimization, machine learning, computer vision, and natural language processing. Some specific applications of metaheuristic algorithms in computer science include(Acan & Ünveren, 2020; Agarwal & Mehta, 2014; Jahwar et al., 2021):

- a. Optimization problems: Metaheuristic algorithms are widely used to solve optimization problems in computer science, such as scheduling, routing, and resource allocation. For example, genetic algorithms can be used to optimize the routing of vehicles in a transportation network(Acan & Ünveren, 2007; Almufti, 2015).
- b. Machine learning: Metaheuristic algorithms can be used to optimize the parameters of machine learning models, such as neural networks and support vector machines. For example, particle swarm optimization can be used to optimize the weights and biases of a neural network for image classification(Rere et al., 2016; Vergin et al., 2015).
- c. Computer vision: Metaheuristic algorithms can be used to solve computer vision problems, such as object detection, segmentation, and tracking. For example, ant colony optimization can be used to optimize the segmentation of objects in an image(Almufti, 2022).
- d. Natural language processing: Metaheuristic algorithms can be used to optimize the performance of natural language processing systems, such as text classification, sentiment analysis, and machine translation. For example, genetic algorithms can be used to optimize the feature selection for text classification(Sun et al., 2009).
- e. Network design: Metaheuristic algorithms can be used to design efficient and robust computer networks, such as wireless sensor networks and ad hoc networks. For example, simulated annealing can be used to optimize the routing and topology of a wireless sensor network(Almufti & Shaban, 2018).

Overall, metaheuristic algorithms have the potential to provide efficient and effective solutions to a wide range of problems as show in figure1-5 and table 1-2, making them a valuable tool in the field of computer science.

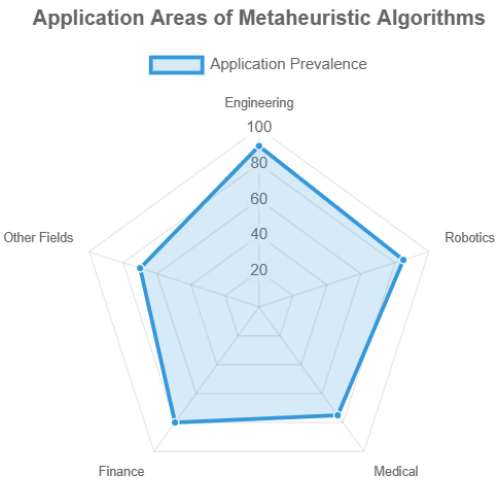


Fig. 1-5:metaheuristics application areas

Table 1-2: Application Domains and Examples

Application Domain	Examples/Use Cases
Engineering	Design optimization, control system calibration, signal processing
Robotics & Swarm Drones	Path planning, task allocation, coordination of swarm robotics
Medical Diagnosis	Feature selection in diagnostic models, optimization in medical imaging
Finance	Portfolio optimization, stock market prediction, fraud detection
Other Fields	Sentiment analysis, weather forecasting, load prediction, spam detection

7.Advantages and Disadvantages of Metaheuristic Algorithms

Metaheuristic algorithms have several advantages and disadvantages, which are important to consider when choosing an optimization algorithm for a particular problem. Here are some of the advantages and disadvantages of metaheuristic algorithms(Abdel-Basset et al., 2018; M. Almufti, 2019):

7.1. Advantages:

- a. Can find good solutions to complex problems: Metaheuristic algorithms are designed to handle complex optimization problems that cannot be solved by traditional optimization methods. They can search large

solution spaces and find good solutions that may not be found by other methods.

- b. Can handle non-linear and non-differentiable problems: Metaheuristic algorithms do not require the objective function to be differentiable, which makes them suitable for non-linear and non-differentiable optimization problems.
- c. Can be parallelized: Metaheuristic algorithms can be easily parallelized, which allows them to exploit the power of modern parallel computing architectures.
- d. Robustness: Metaheuristic algorithms are generally robust to noise and uncertainties in the optimization problem. They can handle noisy or incomplete data, and they are not sensitive to the initial conditions.

7.2. Disadvantages:

- a. No guarantee of optimality: Metaheuristic algorithms are not guaranteed to find the optimal solution to an optimization problem. They may find a good solution, but not the best possible solution.
- b. Computationally expensive: Metaheuristic algorithms can be computationally expensive, especially for large-scale optimization problems. They may require a large number of function evaluations to find a good solution.
- c. Tuning parameters: Metaheuristic algorithms have several parameters that need to be tuned to achieve good performance. Finding the optimal values for these parameters can be a difficult task.
- d. Black-box approach: Metaheuristic algorithms do not provide insights into the underlying structure of the optimization problem. They are a black-box approach that searches for good solutions without providing information on how the solutions were found.

Table 1-3: Comparison of Strengths and Weaknesses

Aspect	Strengths	Weaknesses
Flexibility	Problem-independent; applicable to various optimization tasks	May require extensive parameter tuning
Search Capability	Capable of exploring high-dimensional spaces and escaping local optima	No guarantee of achieving global optimum
Computational Efficiency	Often delivers sufficiently good solutions quickly	Can be computationally heavy if not optimized
Implementation Simplicity	Conceptually simple and adaptable	Stochastic nature can lead to inconsistent outcomes

In conclusion, metaheuristic algorithms have several advantages and disadvantages, and the choice of algorithm depends on the characteristics of the optimization problem and

the available computing resources. Metaheuristic algorithms can provide an effective tool for solving complex optimization problems, but their limitations should also be considered as can be seen in table 1-3.

8. Historical overview of metaheuristic algorithm

The history of metaheuristic algorithms can be traced back to the mid-19th century, when researchers began to develop optimization algorithms that were inspired by natural systems (Almufti, 2019). The graph in Fig. 1-6 illustrates the historical development of metaheuristic algorithms across different decades, showing a steady rise in research activity from the 1990s onward. Evolution-based and swarm intelligence algorithms exhibit the highest growth, particularly after 2000, followed by a surge in hybrid approaches. Physics-based and human-related algorithms show consistent but comparatively moderate growth, highlighting the increasing diversification and hybridization of metaheuristic strategies in recent years.

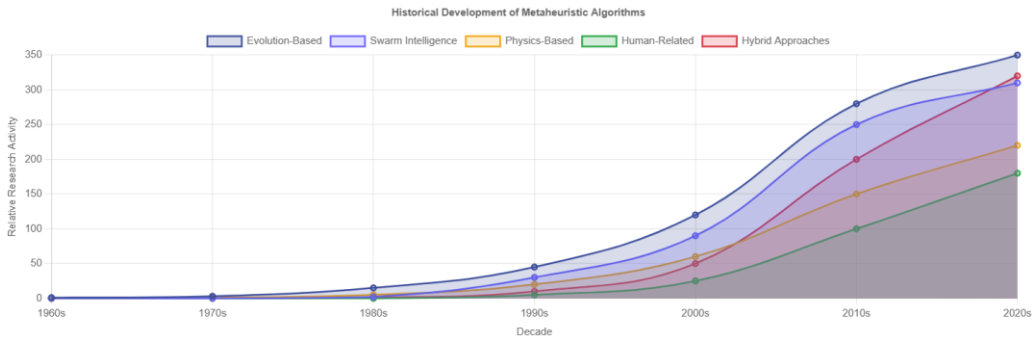


Fig. 1-6: Historical Development of metaheuristics algorithm

In 1953, George Dantzig developed the simplex algorithm, which is a linear programming algorithm that has been widely used in optimization problems. In the 1960s, researchers began to develop optimization algorithms that were inspired by natural systems, such as genetic algorithms, simulated annealing, and tabu search.

In 1975, John Holland published his seminal book "Adaptation in Natural and Artificial Systems", which introduced the concept of genetic algorithms. Genetic algorithms are optimization algorithms that are based on the principle of natural selection, and they are inspired by the process of evolution in biological systems. Genetic algorithms have been widely used in optimization problems, such as scheduling, routing, and resource allocation.

In 1983, Kirkpatrick, Gelatt, and Vecchi developed simulated annealing, which is a stochastic optimization algorithm that is inspired by the process of annealing in metallurgy. Simulated annealing has been widely used in optimization problems, such as traveling salesman problem and protein folding(Gogna & Tayal, 2013; M. Almufti,

2019).

In the 1990s, researchers began to develop optimization algorithms that were based on swarm intelligence, such as particle swarm optimization, ant colony optimization, and bee colony optimization. Swarm intelligence algorithms are inspired by the collective behaviour of social organisms, such as ants, bees, and birds. Swarm intelligence algorithms have been widely used in optimization problems, such as feature selection, data clustering, and routing.

In the 2000s, researchers began to develop optimization algorithms that were based on machine learning, such as reinforcement learning and deep learning. Machine learning algorithms are used to learn from data and make predictions or decisions based on the learned patterns. Machine learning algorithms have been widely used in optimization problems, such as game playing, robotics, and control systems(Shivan Othman et al., 2020).

In recent years, metaheuristic algorithms have become increasingly popular due to their ability to solve complex optimization problems in various fields. The development of high-performance computing and parallel processing has made it possible to apply metaheuristic algorithms to large- scale optimization problems(Acan & Ünveren, 2020; Sherinov et al., 2018).

Table 1-4, is a chronological catalogue of metaheuristic algorithms and their origins, sorted by time periods and authors. It includes foundational techniques, nature-inspired algorithms, and hybrid or advanced variants developed across several decades.

Table 1-4: Metaheuristics algorithms

#	Year	Algorithm	Author
1.	1949	Monte Carlo Method	Metropolis and Ulam
2.	1961	Pattern Search	Hooke and Jeeves
3.	1965	Nelder-Mead Method	Nelder and Mead
4.	1966	Evolutionary Programming	Fogel
5.	1967	Hill Climbing	Pohl
6.	1975	Genetic Algorithm	Holland
7.	1983	Simulated Annealing	Kirkpatrick et al.
8.	1986	Tabu Search	Glover
9.	1987	Niching Genetic Algorithm	Gol and Richardson
10.	1989	Simulated Quenching	Ingber
11.	1989	Search Via Simulated Annealing	Ingber
12.	1989	Memetic Algorithm	Moscato
13.	1991	Ant Colony Optimization	Dorigo et al.
14.	1992	Genetic Programming	Koza
15.	1992	Artificial Life Algorithm	Beer
16.	1994	Cultural Algorithm	Reynolds

17.	1994	Cooperative Coevolution	Potter and De Jong
18.	1994	Cultural Algorithm	Reynolds
19.	1995	Particle Swarm Optimization	Kennedy and Eberhart
20.	1995	Immune Algorithm	Dasgupta and Yu
21.	1995	Simulated Binary Crossover	Deb and Agrawal
22.	1997	Cross-Entropy Method	Reuven Rubinstein
23.	1997	Differential Evolution	Storn and Price
24.	1997	Scatter Search	Glover and Laguna
25.	1998	Hybrid Particle Swarm Optimization Algorithm	Shi and Eberhart
26.	1999	Multi-Objective Genetic Algorithm	Deb
27.	2001	Cooperative Particle Swarm Optimization	Kennedy et al.
28.	2001	Harmony Search	Geem et al.
29.	2002	Estimation of Distribution Algorithm	Pelikan et al.
30.	2002	Particle Swarm Optimization with Mutation Operator	Clerc and Kennedy
31.	2002	Bacterial Foraging Optimization Algorithm	Passino
32.	2002	Quantum-Behaved Particle Swarm Optimization	Kennedy and Mendes
33.	2003	Electromagnetism-Like Algorithm	Birbil and Fang
34.	2004	Hybrid Taguchi-Genetic Algorithm	Ho and Yang
35.	2005	Bee Algorithm	Pham et al.
36.	2005	Artificial Fish Swarm Algorithm	Li and He
37.	2005	Memetic Differential Evolution	Liang et al.
38.	2005	A Hybrid Genetic Algorithm for the Quadratic Assignment Problem	Hao and Moon
39.	2006	Cat Swarm Optimization	Chu, Tsai, and Pan
40.	2007	Artificial Bee Colony Algorithm	Karaboga and Basturk
41.	2007	Imperialist Competitive Algorithm	Atashpaz-Gargari and Lucas
42.	2007	Scatter Search	Glover and Laguna
43.	2007	Enhanced Differential Evolution	Zhang and Sanderson
44.	2008	Firefly Algorithm	Yang
45.	2009	Cuckoo Search	Yang and Deb
46.	2009	Gravitational Search Algorithm	Rashedi et al.
47.	2009	Improved Harmony Search Algorithm	Zhang and Liu
48.	2009	Harmony Search Algorithm with Mutation Operator	Liu et al.
49.	2009	Biogeography-Based Optimization Algorithm	Simon
50.	2010	Bat Algorithm	Yang
51.	2011	Teaching-Learning-Based Optimization	Rao et al.
52.	2011	Krill Herd Algorithm	Gandomi et al.
53.	2011	Multi-Objective Differential Evolution	Das and Suganthan
54.	2011	Big Bang-Big Crunch Algorithm	Esmat et al.
55.	2011	Chaotic Genetic Algorithm	Karaboga and Kaya
56.	2012	Differential Search Algorithm	Civicioglu
57.	2012	Lion Algorithm	Rajakumar B.

58.	2013	Harmony Search Algorithm with Exponential Function	Gandomi et al.
59.	2013	Social Spider Optimization	Erik Cuevas et al.
60.	2014	Magnetic Optimization Algorithm	Ghorbani and Yusof
61.	2014	Fish School Search Algorithm	Bastos Filho et al.
62.	2014	Brain Storm Optimization Algorithm	Zhao et al.
63.	2014	Grey Wolf Optimizer	Mirjalili et al.
64.	2015	Plant Propagation Algorithm	Mohammadi-Ivatloo et al
65.	2015	Water Cycle Algorithm	Abdi et al.
66.	2015	Monkey Algorithm	Kaveh and Talatahari
67.	2015	Black Hole Algorithm	Karaboga and Basturk
68.	2015	U-Turning Ant Colony Optimization	Saman M. Almufti
69.	2016	Whale Optimization Algorithm	Mirjalili and Lewis
70.	2016	Dragonfly Algorithm	Mirjalili et al.
71.	2016	Genetic Algorithm with Dual-Population	Wang et al.
72.	2017	Vibrating Particles System	Kaveh and Ilchi Ghazaan
73.	2020	Multi-objective Dragonfly Algorithm	Niknam et al.
74.	2020	Cuckoo Search with Differential Evolution and Simulated Annealing	Baris and Akay
75.	2020	Hybrid Firefly Algorithm with Simulated Annealing	Baris and Akay
76.	2020	Biogeography-Based Global Optimization Algorithm	Simon
77.	2020	Fuzzy AdaptiveImperialist Competitive Algorithm	Salehi et al.
78.	2020	Multi-objective Elephant Herding Optimization Algorithm	Lashkari and Tavakkoli-Moghaddam
79.	2020	Symbiotic Organisms Search Algorithm	Heidari and Mirjalili
80.	2021	Pareto Local Search with Tabu and Line-Search Techniques	Almohallami and Zalzala
81.	2021	Invasive Weed Optimization with Opposition- Based Learning	Mohan et al.
82.	2021	Social Spider Optimization Algorithm	Dong et al.
83.	2022	Fire Hawk Optimizer	Azizi, Mahdi et al.

9. Conclusion

In conclusion, the history of metaheuristic algorithms spans several decades and involves the development of various optimization algorithms that are inspired by natural systems. Metaheuristic algorithms have become a valuable tool in solving complex optimization problems in various fields, and they are likely to continue to play an important role in the development of new technologies and applications.

Generally, metaheuristics algorithms have been applied to a wide range of scientific, engineering, finance, logistics, Healthcare problems and have demonstrated their effectiveness in finding near-optimal solutions. The use of metaheuristics algorithms is

expected to become even more prevalent in engineering as the complexity and scale of optimization problems continue to increase.

In the real life the daily problems are becoming more and more complex in a way that it become very difficult for a traditional method to solve them within a reasonable time. Metaheuristics algorithm have been used to solve the real-life problems in an optimal time and effort. In the past many algorithms have been developed that belongs to metaheuristic algorithms. This paper is an attempt to provide a historical list of some of metaheuristic algorithms that have been used between 1961 and 2023, it provides the year of establishments, authors name, abbreviations and the reference of the algorithm.

10. References

- Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Metaheuristic Algorithms: A Comprehensive Review. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* (pp. 185–231). Elsevier. <https://doi.org/10.1016/B978-0-12-813314-9.00010-4>
- Acan, A., Altincay, H., Tekol, Y., & Unveren, A. (n.d.). A genetic algorithm with multiple crossover operators for optimal frequency assignment problem. The 2003 Congress on Evolutionary Computation, 2003. CEC '03., 256–263. <https://doi.org/10.1109/CEC.2003.1299583>
- Acan, A., & Unveren, A. (2007). A shared-memory ACO+GA hybrid for combinatorial optimization. 2007 IEEE Congress on Evolutionary Computation, 2078–2085. <https://doi.org/10.1109/CEC.2007.4424729>
- Acan, A., & Ünveren, A. (2015). A great deluge and tabu search hybrid with two-stage memory support for quadratic assignment problem. *Applied Soft Computing*, 36, 185–203. <https://doi.org/10.1016/j.asoc.2015.06.061>
- Acan, A., & Ünveren, A. (2020). Multiobjective great deluge algorithm with two-stage archive support. *Engineering Applications of Artificial Intelligence*, 87, 103239. <https://doi.org/10.1016/j.engappai.2019.103239>
- Agarwal, P., & Mehta, S. (2014). Nature-Inspired Algorithms: State-of-Art, Problems and Prospects. In *International Journal of Computer Applications* (Vol. 100, Issue 14).
- Agrawal, P., Abutarboush, H. F., Ganesh, T., & Mohamed, A. W. (2021). Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009-2019). *IEEE Access*, 9, 26766–26791. <https://doi.org/10.1109/ACCESS.2021.3056407>
- Almufti, S., Marqas, R., & Asaad, R. (2019). Comparative study between elephant herding optimization (EHO) and U-turning ant colony optimization (U-TACO) in solving symmetric traveling salesman problem (STSP). *Journal Of Advanced Computer Science & Technology*, 8(2), 32.
- Asaad, R. R., & Abdalnabi, N. L. (2018). Using Local Searches Algorithms with Ant Colony Optimization for the Solution of TSP Problems. *Academic Journal of Nawroz University*, 7(3), 1–6. <https://doi.org/10.25007/ajnu.v7n3a193>
- Almufti, S. (2017). Using Swarm Intelligence for solving NPHard Problems. *Academic Journal of Nawroz University*, 6(3), 46–50. <https://doi.org/10.25007/ajnu.v6n3a78>

- Almufti, S., Asaad, R., & Salim, B. (2018). Review on elephant herding optimization algorithm performance in solving optimization problems. *International Journal of Engineering & Technology*, 7, 6109-6114.
- Almufti, S. (2021). The novel Social Spider Optimization Algorithm: Overview, Modifications, and Applications. *ICONTECH INTERNATIONAL JOURNAL*, 5(2), 32–51. <https://doi.org/10.46291/icontechvol5iss2pp32-51>
- Almufti, S. (2022). Vibrating Particles System Algorithm: Overview, Modifications and Applications. *ICONTECH INTERNATIONAL JOURNAL*, 6(3), 1–11. <https://doi.org/10.46291/icontechvol6iss3pp1-11>
- Almufti, S. M. (2015). U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem.
- Almufti, S. M. (2022a). Lion algorithm: Overview, modifications and applications *E I N F O. International Research Journal of Science*, 2(2), 176–186. <https://doi.org/10.5281/zenodo.6973555>
- Almufti, S. M. (2022b). Vibrating Particles System Algorithm performance in solving Constrained Optimization Problem. *Academic Journal of Nawroz University*, 11(3), 231–242. <https://doi.org/10.25007/ajnu.v11n3a1499>
- Almufti, S. M., Ahmad, H. B., Marqas, R. B., & Asaad, R. R. (2021). Grey wolf optimizer: Overview, modifications and applications. *International Research Journal of Science, Technology, Education, and Management*, 1(1), 1-1.
- Almufti, S. M., Alkurdi, A. A. H., & Khoursheed, E. A. (2022). Artificial Bee Colony Algorithm Performances in Solving Constraint-Based Optimization Problem. 21, 2022.
- Almufti, S. M., Asaad, R. R., & Salim, B. W. (2018). Review on Elephant Herding Optimization Algorithm Performance in Solving Optimization Problems. *Article in International Journal of Engineering and Technology*, 7(4), 6109–6114. <https://doi.org/10.14419/ijet.v7i4.23127>
- Almufti, S. M., Saeed, V. A., & Marqas, R. B. (2019). Taxonomy of Bio-Inspired Optimization Algorithms.
- Almufti, S. M., & Shaban, A. (2018). U-Turning Ant Colony Algorithm for Solving Symmetric Traveling Salesman Problem. *Academic Journal of Nawroz University*, 7(4), 45. <https://doi.org/10.25007/ajnu.v7n4a270>
- Bäck, T., & Schwefel, H.-P. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1(1), 1–23. <https://doi.org/10.1162/evco.1993.1.1.1>
- Ihsan, R. R., Almufti, S. M., Ormani, B. M., Asaad, R. R., & Marqas, R. B. (2021). A survey on Cat Swarm Optimization algorithm. *Asian J. Res. Comput. Sci*, 10, 22-32.
- Bartz-Beielstein, T., Branke, J., Mehnen, J., & Mersmann, O. (2014). Evolutionary Algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(3), 178–195. <https://doi.org/10.1002/widm.1124>
- Bhuvaneswari, M., Hariraman, S., Anantharaj, B., Balaji, N., & Professor, A. (2014). Nature Inspired Algorithms: A Review. In *International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)* (Vol. 12).
- Dehghani, M., Trojovská, E., & Trojovský, P. (2022). A new human-based metaheuristic algorithm for solving optimization problems on the base of simulation of driving training process. *Scientific Reports*, 12(1), 9924. <https://doi.org/10.1038/s41598-022-14225-7>

- Dhiman, G., & Kumar, V. (2017). Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*, 114, 48–70. <https://doi.org/10.1016/j.advengsoft.2017.05.014>
- Dubey, H. M., Panigrahi, B. K., & Pandit, M. (2014). Bio-inspired optimisation for economic load dispatch: A review. *International Journal of Bio-Inspired Computation*, 6(1), 7–21. <https://doi.org/10.1504/IJBIC.2014.059967>
- Ridwan B. Marqas, Saman M. Almufti, Pawan Shivan Othman, & Chyavan Mohammed Abdulrahman. (2020). Evaluation of EHO, U-TACO and TS Metaheuristics algorithms in Solving TSP. *JOURNAL OF XI'AN UNIVERSITY OF ARCHITECTURE & TECHNOLOGY*, XII(IV). <https://doi.org/10.37896/jxat12.04/1062>
- Fister, I., Yang, X.-S., Fister, I., Brest, J., & Fister, D. (2013). A Brief Review of Nature-Inspired Algorithms for Optimization. <http://arxiv.org/abs/1307.4186>
- Gogna, A., & Tayal, A. (2013). Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*, 25(4), 503–526. <https://doi.org/10.1080/0952813X.2013.782347>
- Kennedy, J., & Eberhart, R. (n.d.). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1942–1948. <https://doi.org/10.1109/ICNN.1995.488968>
- Ferinia, R., Kumar, D.L.S., Kumar, B.S. et al. Factors determining customers desire to analyse supply chain management in intelligent IoT. *J Comb Optim* 45, 72 (2023). <https://doi.org/10.1007/s10878-023- 01007-8>
- Klau, G. W., Lesh, N., Marks, J., & Mitzenmacher, M. (2010). Human-guided search. *Journal of Heuristics*, 16(3), 289–310. <https://doi.org/10.1007/s10732-009-9107-5>
- Luis, J., & Sequera, C. (n.d.). 7 Tune Up of a Genetic Algorithm to Group Documentary Collections. www.intechopen.com
- Almufti, S. (2019). Historical survey on metaheuristics algorithms. *International Journal of Scientific World*, 7(1), 1. <https://doi.org/10.14419/ijsw.v7i1.29497>
- Almufti, S. M. (2022). Hybridizing Ant Colony Optimization Algorithm for Optimizing Edge-Detector Techniques. *Academic Journal of Nawroz University*, 11(2), 135–145. <https://doi.org/10.25007/ajnu.v11n2a1320>
- Almufti, S., Yahya Zebari, A., & Khalid Omer, H. (2019). A comparative study of particle swarm optimization and genetic algorithm. *Journal of Advanced Computer Science & Technology*, 8(2), 40. <https://doi.org/10.14419/jacst.v8i2.29401>
- Shaban, A. A., Dela Fuente, J. A., Salih, M. S., & Ali, R. I. (2023). Review of swarm intelligence for solving symmetric traveling salesman problem. *Qubahan Academic Journal*, 3(2), 10–27.

Chapter 2: Ant Colony Optimization Algorithm

1. Introduction

Ant Colony Optimization algorithm (ACO) is a swarm intelligence (SI) field method that was first represented in 1992 by Marco Dorigo (Dorigo, 1992) in his PhD thesis as a nature-inspired metaheuristic for solving hard combinatorial optimization (CO) problems. Later in 1997, Dorigo and Gambardella published a developed algorithm Ant Colony System (ACS) (Dorigo & Gambardella, 1997). One year later, in 1998, Dorigo presented Ant Colony Optimization algorithms (ACO) for the first time in a conference (Asaad & Abdunabi, 2018). Ant system methods generally and specifically Ant colony optimization over 30 years ago from 1992 to 2022 has been adapted to solve many optimization problem in real life situations (Almufti,2022a). This paper is an attempt to review the ACO algorithm and its proposed applications which help the researcher in future to apply it in other new and promising fields of application such as path planning and path following control of autonomous underwater vehicles (Blum, 2005).

Swarm intelligence works on two basic principles: self- organization and stigmergy see in Fig. 2-1.

- **Self-organization:** This can be characterized by three parameters like structure, multi stability and state transitions. In swarms, interpreted the self-organization through four characteristics: (i) positive feedback,(ii) negative feedback, (iii) fluctuations, and (iv) multiple interactions.
- **Stigmergy:** It means stimulation by work. Stigmergy is based on three principles:(i) work as a behavioural response to the environmental state, (ii) an environment that serves as a work state memory(iii)work that does not depend on specific agents (Almufti, 2015)..

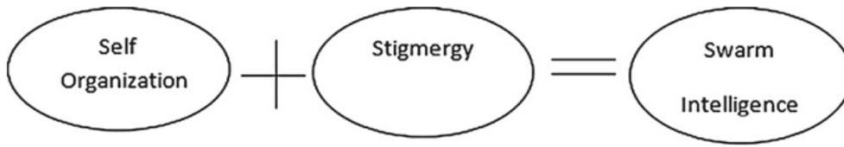


Fig. 2-1: Swarm intelligence basic principles

2. Ant Colony Optimization (ACO)

Marco Dorigo and colleagues introduced the first ACO algorithms in 1992 (Almufti, 2015). The development of these algorithms was inspired by the observation of ant colonies. Ants are social insects. They live in colonies and their behaviour is governed by the goal of colony survival rather than being focused on the survival of individuals (Dorigo, 1992). The behaviour that provided the inspiration for ACO is the ants' foraging behaviour, and in particular, how ants can find shortest paths between food sources and their nest. When searching for food, ants initially explore the area surrounding their nest in a random manner. While moving, ants leave a chemical pheromone trail on the ground. Ants can smell pheromone. When choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food. The pheromone trails will guide other ants to the food source. It has been shown in (Almufti et al., 2021) that the indirect communication between the ants via pheromone trails known as stigmergy enables them to find shortest paths between their nest and food sources. This is explained in an idealized setting in Fig. 2-2.

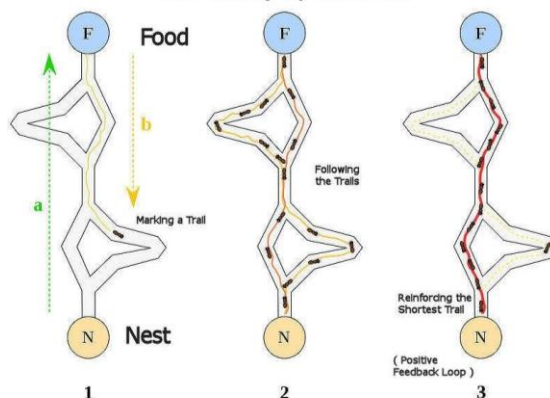


Fig. 2-2: Ant searching behaviour

The traveling salesman problem (TSP) is an important factor in testing and improving the Ant System (AS) algorithms and algorithms that based on the behaviour of Ant

(Almufti, 2015). The first appearance of Ant System by Marco Dorigo in 1992 was tested on the travelling salesman problem (Dorigo, 1992).

3. Travelling Salesman Problem (TSP)

TSP is an NP-hard problem in combinatorial optimization (Almufti et al., 2021). Given a set of cities in which every city must be visited once only and return to the starting city for completing a tour such that the length of the tour is the shortest among all possible tours [1, 2]. In general there are two different kinds of TSP, the Symmetric TSP (STSP) and the Asymmetric TSP (ATSP). the number of tours in the ATSP is $(n-1)!$, Whereas it is $(n-1)!/2$ in STSP for n cities. Formally, the TSP is a complete weighted graph $G(N, A)$ where N is the set of cities which must be visits, and $A(i, j)$ is the set of arcs connecting the cities together (Almufti, 2015). The length between city A_i and A_j can be represented as d_{ij} . Thus the optimal (minimum length) tour to the TSP can be found as shown below in Eq.1.

$$Btour = \left(\sum_{i=1}^{n-1} d_{p(i) p(i+1)} \right) + d_{p(n) p(1)} \quad (1)$$

Where p is a probability list of cities with minimum distance between city (p_i and p_{i+1}).

4. ACO algorithm

ACO algorithm uses for TSP an agent (Artificial ant) which is equal to the number of cities of TSP ($M = N$, M number of ant for N city), each ant makes a solution tour, initially all ants locate at a cities randomly, then every ant chose the next city upon a probability based function depend on both pheromone trail accumulated on edge and heuristic value, the formula used for choosing next city called random proportional rule (2), shows the probability of ant (k) that locate at the city (i) to visit city (j).

$$P_{ij}^k = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{1 \in \mathcal{N}_i^k} [\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}, \text{ if } j \in \mathcal{N}_i^k \quad (2)$$

Where (α and β) are variables that determine the influence of pheromone trail and heuristic information, where α is a parameter to regulate the influence of τ_{ij} and β is a parameter to regulate the influence of η_{ij} , (α and β) have effect in the process of choosing next city as shown below.

$\alpha < \beta$ The closet city is more likely to be chosen.

$\alpha > \beta$ The arcs that have more pheromone intensity are more likely to be use.

$\alpha = 0$, heuristic value uses for the choosing without pheromone.

$\beta = 0$, pheromone uses for the choosing without heuristic value, which may cause poor

result.

$$\eta_{ij} = 1/d_{ij} \quad (3)$$

(η_{ij}) is a priori available heuristic value, (d_{ij}) is the distance between cities (i and j), (τ_{ij}) is the pheromone trail matrix, and ($\mathcal{N}_i^k$) is the set of the neighborhood that ant (k) haven't visit them yet.

The probability that ant (k) choose arc (i, j) increase with the value pheromone trail (τ_{ij}) and heuristic value (η_{ij}).

In this thesis we uses ($\alpha = 1$ and $\beta = 2$), the initial value of pheromone matrix usually set to a small value which is greater than zero.

$$\text{for all } (i, j) \Leftrightarrow \tau_{ij} = \tau_0, \text{ where } \tau_0 > 0 \quad (4)$$

Ants chose the city which contain larger amount of pheromone on the connecting edges between the current city and next one.

Each ant after adding new city to its tour makes a local pheromone update (5), by adding the $\Delta\tau_{ij}$ to the old τ_{ij}

$$\tau_{ij}^{new} = \tau_{ij}^{old} + \Delta\tau_{ij} \quad (5)$$

Where the value of $\Delta\tau_{ij}$ can be found as shown in (6), $\Delta\tau_{ij}$ changes according to the length (L_k) tour (7)

$$\Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k \quad (6)$$

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{if ant } k \text{ uses arc } (ij) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Where τ_{ij} is the pheromone trail on the arc (ij), m is number of ants, and $\Delta\tau_{ij}$ is the summation of change in the $\Delta\tau_{ij}^k$, which is equal to a constant value (Q) over current length of the tour constructed by ant (k), in this thesis we use value ($Q=1$). The process of pheromone trail update continues until all ant complete their tour by visiting all cities and returning to the start city, after all ants terminate the construction of tour it follow by pheromone evaporation (8) and a global pheromone update (4) for the shortest tour. The shortest tour will have the greatest amount of pheromone. Pheromone evaporation reduces the amount of pheromone in an arc which makes ant to follow one path.

$$\tau_{ij}^{new} = (1 - \psi) \cdot \tau_{ij}^{old} \quad (8)$$

Whereas ($0 < \psi < 1$) is a constant quantity used for reducing pheromone trail, here we used ($\psi = 0.5$) that decrease τ_{ij} value by 0.5 which iteratively lead to disappear pheromone trail in unused arc. Fig 2-3 shows ACO Pseudo-code and Fig. 2-4 shows ACO flowchart.

Initialize

Loop

Each ant is positioned on a starting node

Loop

***Each ant applies a state transition rule to incrementally
build a solution and a local pheromone updating rule***

Until all ants have built a complete solution

A global pheromone updating rule is applied

Until end condition

Fig. 2-3: ACO Pseudo-code

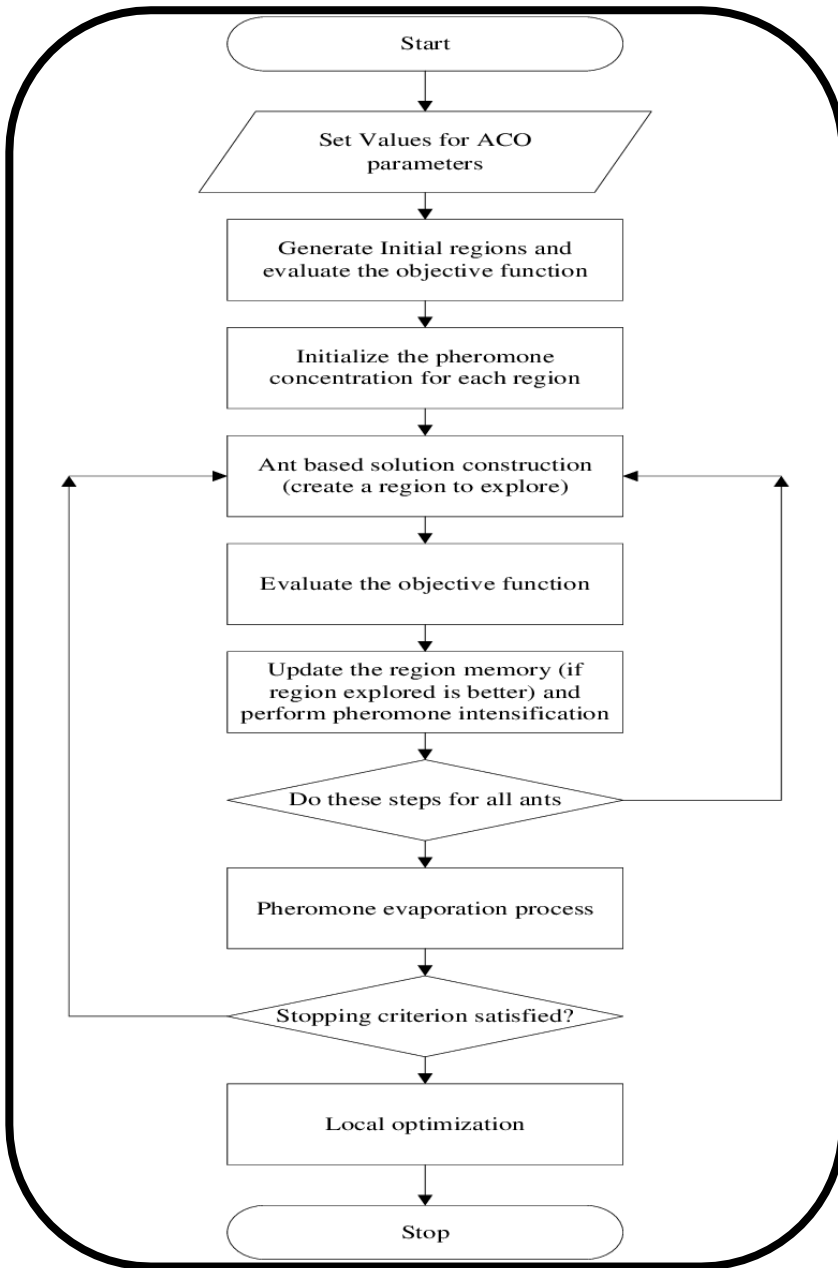


Fig. 2-4: ACO flowchart

5. Modifications of Ant based optimization algorithm

Generally, all the metaheuristics algorithms after its first appearance undergoes many modifications and improvements, so that it can be used to solve various problems (Almufti, 2019). During 30 years from the first appearance of Ant based optimization algorithm in 1992. To meet the demands of the real-world problems many modifications has been applied to the original ACO to improve the performances of the proposed

algorithm. In this section, some of the modifications and improvements of the ACO algorithm are listed and arranged according to the development year, as shown bellow in Table 2-1.

Table 2-1: Ant based optimization algorithm modifications

Year	Scientist Name	Modification
1992	M. Dorigo	Proposed the ant system in his doctoral thesis (which was published in 1992). A technical report extracted from the thesis and co-authored by V. Maniezzo and A. Colorni was published five years later.
1995	Gambardella and Dorigo	Proposed ant-q, the preliminary version of ant colony system as first extension of ant system.
1996	Gambardella and Dorigo	Proposed ant colony system.
1997	Dorigo and Gambardella	Proposed ant colony system hybridized with local search.
1997	Schoonderwoerd and his colleagues	Published an improved application to telecommunication networks.
1998	Dorigo	Dorigo launches first conference dedicated to the ACO algorithms.
1998	Stützle	Proposes initial parallel implementations.
1999	Gambardella, Taillard and Agazzi	Proposed macs-vrptw, first multi ant colony system applied to vehicle routing problems with time windows.
1999	Bonabeau, Dorigo and Theraulaz	Publish a book dealing mainly with artificial ants.
2000	M. Dorigo , G. Di Caro et T. Stützle,	Special issue of the Future Generation Computer Systems journal on ant algorithms.
2000	Hoos and Stützle	Invent the max-min ant system.
2000	Gutjahr	Provides the first evidence of convergence for an algorithm of ant colonies.
2001	Iredi and his colleagues	Published the first multi-objective algorithm.
2002	Bianchi and her colleagues	Suggested the first algorithm for stochastic problem.
2004	Dorigo and Stützle	Publish the Ant Colony Optimization book with MIT Press.
2004	Zlochin and Dorigo	Show that some algorithms are equivalent to the stochastic gradient descent, the cross-entropy method and algorithms to estimate distribution.

2005	M. Zlochin, M. Birattari, N. Meuleau, et M. Dorigo	First applications to protein folding problems.
2012	Prabhakar and colleagues	Publish research relating to the operation of individual ants communicating in tandem without pheromones, mirroring the principles of computer network organization. The communication model has been compared to the Transmission Control Protocol.
2015	Saman M. Almufti	Proposed U-TACO powered by GDA and 2- OPT for solving TSP.
2016	Zaidman, Daniel; Wolfson, Haim J	First application to peptide sequence design.
2017	Mladineo, Marko; Veza, Ivica; Gjeldum, Nikola	Successful integration of the multi-criteria decision-making method PROMETHEE into the ACO algorithm (HUMANT algorithm).
2022	Saman M. Almufti	Combined ACO with Canny for detecting image edges.

6. Ant based algorithms Applications

Ant system methods generally and specifically Ant colony optimization over years ago has been adapted to solve many optimization problem in real life situations such as Mobile and Wireless Sensor Networks (WSNs), vehicle routing problem, Network Security, Computer Architecture, Data Mining and many other problems (Stutzle, Lopez-Ibznéz, & Dorigo, 2011), (table 2-2) shows some applications of ant system in general and ACO in various field.

Table 4-1: Summarize GWO applications in different field

Problem/Application	Algorithm Name	Authors
Traveling salesman	AS	Dorigo, Maniezzo & Colomi
	Ant-Q	Gambardella & Dorigo
	ACS & ACS-3-opt	Dorigo & Gambardella
	MMAS	Stitzle & Hoos
	ASrank	Bullnheimer, Hartl & Strauss
	BWAS	Cordon, et al.
	U-TACO	Saman M. Almufti
Quadratic assignment	AS-QAP	Maniezzo, Colomi & Dorigo
	HAS-QAPa	Gambardella, Taillard & Dorigo
	MM AS-QAP	Stitzle & Hoos
	ANTS-QAP	Maniezzo
	AS-QAPb	Maniezzo & Colomi
Scheduling problems	AS-JSP	Colomi, Dorigo & Maniezzo
	AS-FSP	Stitzle
	AC S-SMTTP	Bauer et al

	AC S-SMTWTP	den Besten, Stitzle & Dorigo
	ACO-RCPS	Merkle, Middendorf & Schneck
Vehicle routing	AS-VRP	Bullnheimer, Hartl & Strauss
	HAS-VRP	Gambardella, Taillard & Agazzi
Connection-oriented network routing	ABC	Schoonderwoerd et al
	ASGA	White, Pagurek & Oppacher
	AntNet-FS	Di Caro & Dorigo
Connection-less network routing	ABC-smart ants	Bonabeau et al
	AntNet & AntNet-FA	Di Caro & Dorigo
	Regular ants	Subramanian, Druschel & Chen
	CAF	Heusse et al
	ABC-backward	van der Put & Rothkrantz
Sequential ordering	HAS-SOP	Gambardella & Dorigo
Graph coloring	ANTCOL	Costa & Hertz
Shortest Common supersequence	AS-SCS	Michel & Middendorf
Frequency assignment	ANTS-FAP	Maniezzo & Carbonaro
Generalized assignment	MMAS-GAP	Ramalhinho Lourenco & Serra
Multiple knapsack	AS-MKP	Leguizamon & Michalewicz
Optical networks routing	ACO-VWP	Navarro Varela & Sinclair
Redundancy allocation	ACO-RAP	Liang & Smith
Constraint satisfaction	Ant-P-solver	Solnon
Image Edge Detection	ACO-Canny	Saman M. Almufti

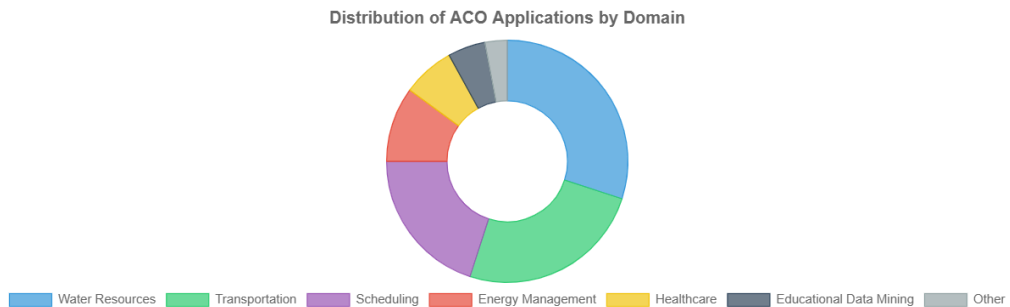


Fig. 2-5: Distribution of ACO Applications by Domain

The donut chart in Fig.2-5 illustrates the diverse areas where Ant Colony Optimization (ACO) has been applied. The largest segments represent its dominant use in water resources and transportation, followed by significant applications in scheduling and energy management. Smaller portions of the chart indicate growing but less prominent adoption in healthcare, educational data mining, and other domains, demonstrating ACO's versatility across both engineering and non-engineering fields.

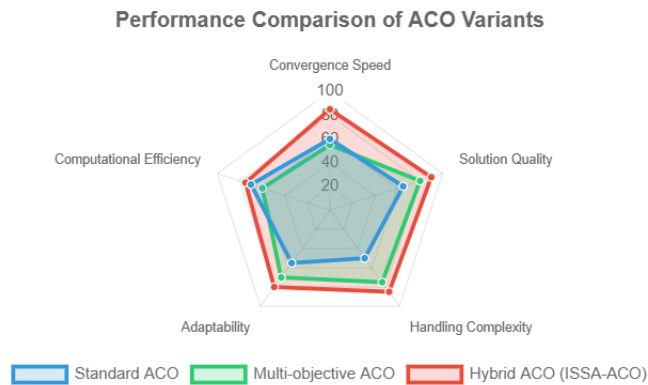


Fig. 2-6: Performance Comparison of ACO Variants

The radar chart in Fig.2-6 compares Standard ACO, Multi-objective ACO, and Hybrid ACO (ISSA-ACO) across five performance criteria: convergence speed, solution quality, handling complexity, adaptability, and computational efficiency. Standard ACO demonstrates balanced but modest capabilities, making it suitable for simpler problems with fewer constraints. Multi-objective ACO shows enhanced adaptability and complexity handling due to its ability to optimize multiple conflicting objectives, though at the cost of slightly reduced computational efficiency. In contrast, the Hybrid ACO variant outperforms both, exhibiting superior performance across all dimensions, particularly excelling in convergence speed and solution quality, due to the synergistic integration of Improved Social Spider Algorithm with ACO. This comparison highlights the clear advantage of hybridized models in solving complex and dynamic optimization problems more effectively.

7. CONCLUSION

Ant system methods belongs to Swarm-based metaheuristic algorithm the first algorithm that used ant behaviours was proposed in 1992 by Marco Dorigo et al. (Almufti, 2022). After its appearance, many modifications has been proposed on it and it has been adapt to solve various problem in different fields. This paper firstly addressed this overviewed the Ant based algorithm and then it presented some of its modifications were presented in detail, finally some of its application considering parameter tuning, different approaches for feature selection and classification, and then hybridized forms were discussed. The applications in different fields were discussed including engineering medical, power dispatch, reliability optimization etc.

8. REFERENCES

Almufti, S. M., Maribojoc, R. P., & Pahuriray, A. V. (2022a). Ant-based system: Overviews, modifications, and applications from 1992 to 2022. *Polaris Global Journal of Scholarly Research and Trends*, 1(1), 10–17. <https://doi.org/10.58429/pgjsrt.v1n1a85>

- Almufti, S.M. (2015). U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem. Famagusta, Cyprus: EMU. Retrieved from <http://i-rep.emu.edu.tr:8080/jspui/handle/11129/1734>
- Almufti, S.M. (2019). Historical survey on metaheuristics algorithms. *International Journal of Scientific World*.
- Almufti, S.M. (2022b). Hybridizing Ant Colony Optimization Algorithm for Optimizing Edge-Detector Techniques. *Academic Journal of Nawroz University*, 11(2), 135-145.
- Almufti, S.M., Marqas, R.B., Othman, P.S., & Sallow, A.B. (2021). Single-based and Population-based Metaheuristics for Solving NP-hard Problems. *Iraqi J Sci*, 62(5), 1-11.
- Asaad, R.R., & Abdalnabi, N.L. (2018). Using local searches algorithms with Ant colony optimization for the solution of TSP problems. *Academic Journal of Nawroz University*, 1-6.
- Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 353–373.
- Dorigo, M. & Gambardella, L.M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 1(1), 53-66.
- Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. Politecnico di Milano, Italy.
- Stutzle, T., Lopez-Ibáñez, M., & Dorigo, M. (2011). A Concise Overview of Applications of Ant Colony Optimization. In *Wiley Encyclopedia of Operations Research and Management Science*.

Chapter 3: Lion Algorithm

1. Introduction

Generally, metaheuristic algorithms describe as a "master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality" (Marqas, Almufti, Ahmed, & Asaad, 2021).

In this chapter, a Metaheuristic algorithm based on lion's behaviours called Lion Algorithm (LA) is introduced. The basic (LA) was firstly introduced by Rajakumar In 2012, it was called as Lion's Algorithm (LA) (Rajakumar B. , 2012), which inspired the raw inspirations of lion's unique social behaviour Mating and Territorial (Rajakumar B. , 2012). But additionally to mating and territorial, lions have many other behaviours such as migration, territorial marking, unique style of prey capturing, roaming, moving to safe place, and other behaviours. So, during the past decent, various algorithms has been proposed based on the basic Loin's Algorithm for solving problems related to numerous fields (Almufti S. M., 2017).

After introducing the basic Loin's Algorithms, this chapter high-light the new Algorithms that inspired the behaviours of Lain's and the fields and application that are used in.

2. Lion's Social Behaviour

In the nature, Lions belongs to cat species and they have an exciting social behaviour to preserve the pried member stronger in every generation. Lions are social animals, unlike most other members of the cat family, living in a pride (family group) with between 20 and 30 members. Some prides have just one male, others up to four (Almufti S. M., 2015). Lions are strongly territorial and will fight off any strange male who tries to enter their territory. A cub (lion child) requires 2-4 years to reach the maturity age, during that time the territorial lion have to defend for the territory. In between these 2-4 years, nomadic lions out of pride may try to attack the pride and leads to a war between the territorial and nomadic lions, which is called territorial defence. The lions that belong to

a pride together work to defeat the nomadic lion. In the defeating agents the nomadic lions, if the territorial lose the war may be either killed or driven out of the pride. The nomadic lion becomes the territorial lion by killing the cubs of old territorial lion. And the new territorial lion forces the female lion on the pride to oestrus and copulate to give birth to their own cubs (Bauer, de, & Silvestre, 2003). After the cubs get matured, they take over the territorial lion a war occur between the old territorial male and new territorial male. If they seem to be stronger than the territorial lion to take over the pride, the territorial lion may be either killed or driven out of the pride.

3. Standard Lion Algorithm (LA)

Rajakumar Presented the basic stander Lion Algorithm as a searching algorithm in 2012 (Rajakumar B. , 2012) as an inspirations of the lion behaviours. Following to this, the algorithm was modified and reconstructed with some improvements. Thus, many other algorithms developed base on the principles of stander lion algorithm. The standard LA passes throw Six steps: (1) pride generation, (2) fertility evaluation, (3) mating, (4) territorial defence, (5) territorial takeover, and (6) termination. As it is illustrated bellow in Fig. 3-1

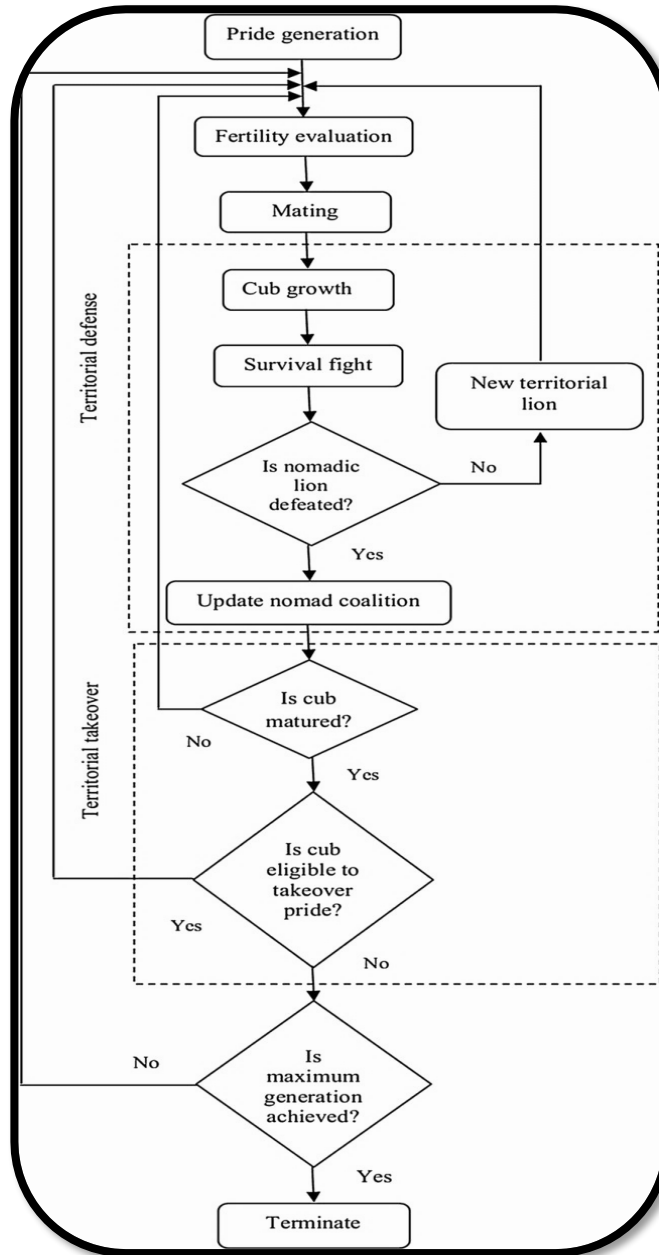


Fig. 3-1: Standard Lion Algorithm flowchart

Pride generation represents the starting step in Lion Algorithm, which is similar to many other swarm-based and evolution-based algorithms, in this step of algorithm all the male, female, nomadic lions and the objective model are initialized. After completing the initialization (Pride generation step) Mating step is started to produce cubs from the birth pride, that includes the periodic fertility evaluation, which is considered as the most remarkable process among the entire processing steps (Rajakumar B. R., 2020). The

incomparable steps from the other optimizations algorithms is the territorial-defense and territorial-takeover steps that shows the social behaviour of pride.

These two processes are accounted as the major function to direct the algorithm in the searching process and so to find the optimal solution. The termination criteria of LA depend on the problem model and therefore that might be processed on the basis of gained solution's optimality or on a count of iterations/generations. The stepwise operation of standard LA is explained by the pseudo-code given in Algorithm 1.

3.1. Pride generation

According to the pride definition and Eq. (1), pride is initialized with a territorial lion Xmale, its lioness Xfemale and a nomadic lion Xnormal. The nomadic lion is not a member of pride, despite its generation is discussed in pride generation process. The lions representation is as similar to the solution vector representation. The vector elements of Xmale, Xfemale and Xnomad, i.e., xmale l, x female l and xnomad l are arbitrary integers within the minimum and maximum limits when $n > 1$ (search with real encoding), where, $l = 1, 2, \dots, L$. Here, L represents the length of the lion that can be determined as follows

$$L = \begin{cases} n; & n > 1 \\ m; & otherwise \end{cases} \quad (1)$$

where, n and m consider two integers that corresponds the length of lions. In case of $n = 1$, the algorithm will search for a binary encoded lion and so the vector-elements are either be generated as 1 or 0, to satisfy the constraints in Eqs. (1) and (3).

$$g(xl) \in (x_l^{min}, x_l^{max}) \quad (2)$$

$$m \% 2 = 0, \quad (3)$$

Where

$$g(x_l) = \sum_{l=1}^L x_l 2^{\left(\frac{L}{2}-l\right)} \quad (4)$$

LA uses Eq. (2) and (4) for ensuring that the generated binary lion is inside the solution space, whereas the Eq. (6) for keeping an equal numbers of binary bits in

both side of the decimal point. The generated Xnomad is placed into one of the two nomadic lion positions. assuming there are two nomadic lions, and they are trying to invade territory. The other lion will only be initialized when it is needed for territorial defence. For now, the position remains null and Xnomad will be represented by Xnomad 1.

3.2. Fertility evaluation

In the sequential process of the lion algorithm, every territorial lion and lioness gradually age or sometimes become infertile. This can make the lion seem slow or sluggish when it comes to fighting for its own survival or taking over territory. If Xmale and Xfemale were to get enough of their fitness, they would either reach global optima or local optima from which they could not lead us to better solutions. Fertility evaluation can help you find local solutions that are optimal for you. In this process, Xmale is found to slow down and its deceleration rate Lr is increased by one if f(Xmale) is greater than fref, which is the reference fitness. When the Lr limit is reached, a territorial defence response occurs (Almufti, 2022),. This is helped along by the fertility rate of the X-female, which is increased by one after crossover. If Sr exceeds the tolerance Smax r, then female X will undergo updating as given in the equation. I think you may be misunderstanding me. I don't think you are understanding me correctly. When the updated female Xfemale+ is considered to be equal to the original female Xfemale, the mating process can be performed. On the contrary, the update continues until the number of female gc generation reaches gmax c. If there is no Xf female + to replace X female throughout the modernization process, it can be determined that X female is still reproductive enough to produce better cubs .

$$x_l^{female+} = \begin{cases} x_l^{female+}; & \text{if } l = k \\ x_l^{female}; & \text{otherwise} \end{cases} \quad (5)$$

$$x_k^{female+} = \min[x_k^{max}, \max(x_k^{min} - \nabla_k)] \quad (6)$$

$$\nabla_k = [x_k^{female} + (0.1r_2 - 0.05)(x_k^{male} - r_1 x_k^{female})] \quad (7)$$

where, x female+ l and x female+ k are the l th and kth vector elements of Xfemale+, respectively, k is a random integer generated within the interval [1, L], ∇ is the female update function, r1 and r2 are random integers generated within the interval [0, 1].

3.3. Mating

In the lion algorithm, mating consists of two main steps and one supplementary step. Crossover and mutation are considered as the primary steps in evolution, while gender clustering is called as the supplementary step. Many works have been written about crossover and mutation operations and their importance for evolution algorithms. These operators motivate us, so we have included them in our algorithm. By crossing different elements between X_{male} and X_{female} , they create cubs that are solutions made up of both X_{male} and X_{female} 's parts. We follow the natural littering rate of four cubs in a lioness pregnancy. This leads to the birth of four cubs (Almufti, 2022).

The proposed crossover operation for generating one cub is illustrated in Fig. 3-2. An X_{cub} of an X_{male} and an X_{female} is the sum of Hadamard product of crossover mask and X_{male} and Hadamard product of complement of the same crossover mask and X_{female} , provided the crossover mask is essentially to be a binary vector with $Cr \cdot L$ number of binary ones. The B mask is varied to create each cub, i.e. The p th B_p mask is used to get $X_{cubs}(p)$. These four cubs also undergo mutation to produce four new cubs, henceforth we use the terms “ X_{cubs} ” to represent the cubs obtained from the cross and “ X_{new} ” to represent the cubs obtained from the mutation. These eight cubs are in the cub pool and are being grouped by gender to create the final X_{m_cub} and X_{f_cub} .

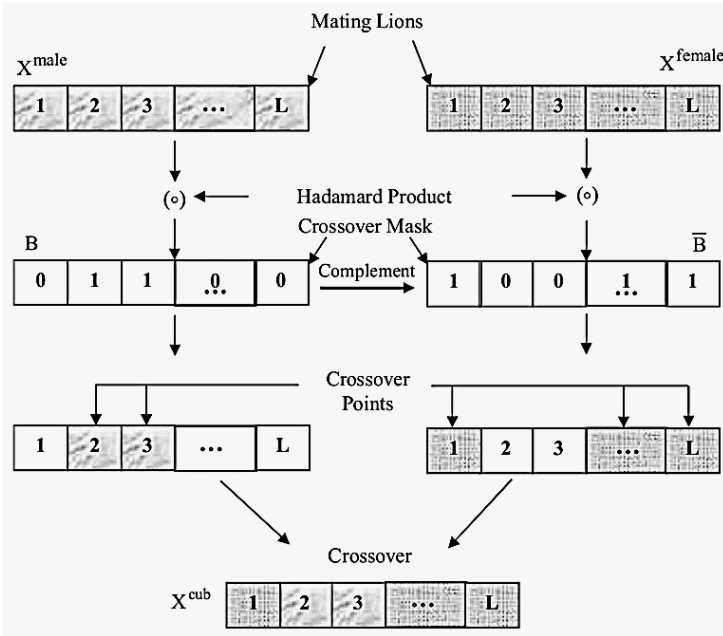


Fig. 3-2: Mating in LA

3.4. Lion operators

Territorial defence not only allows for a broad search of the solution space, but also helps the algorithm in avoiding the local optimal point and recognizing multiple solutions with equal fitness. The territorial defence can be divided into three stages: forming a nomad coalition (Yazdani & Jolai, 2016), fighting for survival and finally updating the nomad coalition. The winner-take-all strategy is used to identify $x^{e-nomad}$, which simplifies the nomad coalition process. Following that, if the requirements in Eqs. (8)–(10) are met, $x^{e-nomad}$ is chosen.

$$f(x^{e-nomad}) < f(x^{male}) \quad (8)$$

$$f(x^{e-nomad}) < f(x^{m_cub}) \quad (9)$$

$$f(x^{e-nomad}) < f(x^{f_cub}) \quad (10)$$

When X_{male} is defeated, pride is updated, but when X_e nomad is defeated, nomad coalition is updated. The pride updating process involves replacing X_{male} with X_e nomad, whereas updating a nomad coalition entails selecting only one X_{nomad} with E_{nomad} greater than or equal to the exponential of unity with the other position being filled only at the time of the next territorial defense. If x^{m_cub} and x^{f_cub} are matured, i.e. when the age of the cubs surpasses the maximum age for cub maturity A_{max} , territorial takeover leads the algorithm to update x^{male} and x^{female} .

3.5. Termination

When at least one of the following two termination criteria is met, the algorithm execution is terminated.

$$N_g > N_g^{Max} \quad (11)$$

$$|f(x^{male}) - f(x^{Optimal})| \leq e_t \quad (12)$$

where, N_g is the number of generations, which is initialized as zero, when a territorial takeover occurs it gradually incremented by one, N_g^{Max} represents the maximum number of generations and error threshold denoted by e_t . In Eq. (12) second criterion can be considered only when the target minimum $f(x^{Optimal})$ (or maximum) is known and $f(x^{Optimal})$ does not mean that $x^{Optimal}$ is known.

4. Standard Lion Algorithm pseudocode

Here is the standard pseudocode for the Lion Algorithm (LA), a nature-inspired metaheuristic that simulates the social and territorial behaviour of lions, including

pride formation, mating, and territorial defence. This algorithm balances exploration through nomadic lions and exploitation via pride-based cooperation and mating.

Step 1.	Initialize Xmale, Xfemale and Xnomad 1
Step 2.	Calculate $f(X_{male})$, $f(X_{female})$ and $f(X_{nomad\ 1})$
Step 3.	Set $f_{ref} = f(X_{male})$ and $N_g = 0$
Step 4.	Store Xmale and $f(X_{male})$
Step 5.	Perform fertility evaluation
Step 6.	Perform mating and obtain cubpool
Step 7.	Perform gender clustering and obtain X_{m_cub} and X_{f_cub}
Step 8.	Initialize Acub as zero
Step 9.	Execute cub growth function
Step 10.	Perform territorial defence; if defence result 0, go to step 4
Step 11.	If $Acub < A_{max}$, go to step 9
Step 12.	Perform territorial takeover and obtain updated Xmale and Xfemale
Step 13.	Increase N_g by one
Step 14.	If the termination criteria are not met, go to step 4, otherwise terminate the process.

5. Modifications of LA

In general, all metaheuristics algorithms undergo several adjustments and enhancements after their initial appearance, so that they can be utilized to tackle a variety of issues (Almufti, 2022; Wen et al., 2015).

After the appearance of standard lion algorithm (LA) in 2012, Many adjustments have been made to the original LA to increase the performance of the suggested algorithm in order to meet the needs of real-world problems.

In this section, some of the modifications and improvements of the LA algorithm are listed and arranged according to the development year (Almufti, 2022), as shown in Table3-1.

Table 3-1: LA based algorithms

#	Abbr.	Name	Author	Year	Ref.
1.	MO- ADDOFL	Multi-objective-based adaptive dynamic directive operative fractional lion algorithm	Satish Chander	2017	(Satish, P., & Praveen, 2017)

2.	ALF-TOHIP	Fractional Lion Topology-Hiding Multipath Routing Protocol	Ambekar Kolekar	2017	(RK & UD, 2017)
3.	FLA	Fractional LA	Satish Chander	2018	(Chander, Vijaya, & Dhyani, 2018)
4.	ADDOFL	Adaptive Dynamic Directive Operative Fractional Lion	Ranjan N, Prasad R	2018	(Ranjan & Prasadb)
5.	LFNN	lion fuzzy neural network-based	Ranjan N, Prasad R	2018	(Ranjan & Prasadb)
6.	KLOA	K-Lion Optimization Algorithm	Jagatheesh kumar	2018	(G & brunda, 2018)
7.	IKLOA	Improved K-Lion Optimization Algorithm	Jagatheesh kumar	2018	(G & brunda, 2018)
8.	M-LionWhale	Multi-Lion whale optimisation algorithm	Chintalapalli & Ananthula	2018	(Chintalapalli & Ananthula, 2018)

6. LA Applications

Over year, Standard Lion Algorithm (La) algorithm and its modifications showed high-performance in solving different real-world problems and it has been used for solving unconstrained, constrained, multi-objective and NP-Hard problems in fields of engineering, medical, environment, etc as summarized in Table 3-2.

Table 3-2: summarize LA applications in different field

#	Application	Discussion	Ref.
1.	nonlinear system identification	The accuracy of the system identification process is improved by using global optimization techniques in nonlinear system identification, specifically bilinear system identification. To achieve precise system characteristics, the LA is used in bilinear system identification. (BR, 2014)	(BR, 2014)
2.	Data clustering	Clustering is a data partitioning technique that groups homogeneous data into one cluster and divides heterogeneous data into interclusters. It's a method of determining the cluster centroid, which represents the whole cluster. It's an optimization problem that can be solved. The Adaptive Dynamic Directive Operative Fractional Lion (ADDOFL) algorithm, a version of LA, has been documented in the literature to select the centroids that can cluster the data more successfully.	(Chander, Vijaya, & Dhyani, 2018)

3. Wireless sensor network (WSN)	in WSN, Clustering of sensor nodes and selection of cluster heads, which are the heads of sensor clusters, are part of the hierarchical routing process. The choice of cluster heads is critical because it determines the network's lifespan and energy efficiency. The cluster head selection problem is handled by fractional lion (FLION) [36], resulting in a longer network lifetime.	(RK & UD, 2017)
4. Feature selection	Feature selection is a technique in data processing that determines the ideal subset of features based on a set of assessment criteria and decreases the number of unneeded characteristics. The best feature selection is required to get the highest level of classification accuracy. LA has successfully addressed this issue by incorporating a greedy search mechanism into territorial defence.	(KC, JC, & JT, 2018)
5. Mobile Ad hoc Network (MANET)	The most well-known method of data transfer is the Mobile Ad hoc Network (MANET). The main problem with MANET is determining the best effective routing path. For better solution search, a modified LA called fractional LA is implemented into the TOHIP (Topology-Hiding Multipath Routing Protocol) protocol and termed AFL-TOHIP (RK & UD, 2017). The M-Lion Whale hybrid optimization technique, which includes the LA into the Whale Optimization Algorithm (WOA) for secure routing, is used to build a programming model (Chintalapalli & Ananthula, 2018)	(RK & UD, 2017), (Chintalapalli & Ananthula, 2018)
6. Text classification	One of the text mining tasks is text classification, which categorizes documents based on their properties and user needs. The dimensionality of the search space is one of the most significant issues in text classification. It is directly affected by the dimensionality curse. The optimization principle is applied here by utilizing the LA to find the optimal weights of a fuzzy neural network. (Prasadb & Ranjan)	(Ranjan & Prasadb)
7. Vehicular Ad hoc Networks (VANETs)	Mobile Ad hoc Networks (MANETs) are stated to include VANETs, which provide reliable road safety. An optimization technique is the optimum answer since the routing process analyses numerous parameters such as Quality of Service (QoS) limitations, congestion parameters, and many more. The LA has been utilized to tackle the route finding problem in VANET because it has been validated for benchmark problems.	(MB & N, 2018)
8. Social network analysis	Social networks are information networks in which people can communicate and share similar interests. In this social network study, community detection is viewed as an optimization problem. As a result, LA is used to assess the amount of user communities that should persist in social networks.	(Y, Y, & M, 2018)

7. Conclusion

Standard Lion Algorithm (La) algorithm is a Swarm-based metaheuristic algorithm proposed in 2012. After its appearance, many modifications have been proposed on it and it has been adapted to solve various problem in different fields. This chapter firstly addressed this overviewed the original LA algorithm and then it presented some of its modifications were presented in detail, finally some of its application considering parameter tuning, different approaches for feature selection and classification, and then hybridized forms were discussed. The applications in different fields were discussed including engineering medical, power dispatch, reliability optimization etc.

8. References

- Almufti, S. M. (2015). U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem. Retrieved from Hdl.handle.net
- Almufti, S. M. (2017a). Historical survey on metaheuristics algorithms. *International Journal of Scientific World*, 7(1), 1-12. doi:<https://doi.org/10.14419/IJSW.V7I1.29497>
- Almufti, S. M. (2017b). Using Swarm Intelligence for solving NP-Hard Problems. *Academic Journal of Nawroz University*, 6(3), 46-50. doi:<https://doi.org/10.25007/ajnu.v6n3a78>
- Bauer, H., de, I. H., & Silvestre, I. (2003). Lion (*Panthera leo*) social behaviour in the West and Central African savannah belt. *Mammalian Biology*, 68(4), 239-243. doi:<https://doi.org/10.1078/1616-5047-00090>
- BR, R. (2014). Lion algorithm for standard and large scale bilinear system identification: a global optimization based on lion's social behavior. *IEEE congress on evolutionary computation (CEC)*, 2116–2123.
- Chander, S., Vijaya, P., & Dhyani, P. (2018). Multi kernel and dynamic fractional lion optimization algorithm for data clustering. *Alexandria Engineering Journal*, 57(1), 267-276. doi:<https://doi.org/10.1016/j.aej.2016.12.013>
- Chintalapalli, R. M., & Ananthula, V. R. (2018). M-LionWhale: multi-objective optimisation model for secure routing in mobile ad-hoc network. *IET Communications*, 12(12), 1406-1415. doi:[10.1049/iet-com.2017.1279](https://doi.org/10.1049/iet-com.2017.1279)
- G, J., & brunda, S. S. (2018). An Improved K-Lion Optimization Algorithm With Feature Selection Methods for Text Document Cluster. *International Journal of Computer Sciences and Engineering*, 6(7), 245-251.
- KC, L., JC, H., & JT, W. (2018). Feature selection with modified lion's algorithms and support vector machine for high-dimensional data. *Appl Soft Comput* 68.
- Marqas, R. B., Almufti, S. M., Ahmed, H. B., & Asaad, R. R. (2021). Grey wolf optimizer: Overview, modifications and applications. *International Research Journal of Science, Technology, Education, and Management*, 1(1), 44-56. doi:<https://doi.org/10.5281/zenodo.5195644>
- MB, W., & N, G. (2018). Route discovery for vehicular Ad hoc networks using modified lion algorithm. *Alexandria Eng J* 57.

- Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*.
- Rajakumar, B. (2012). The Lion's Algorithm: A New Nature-Inspired Search Algorithm. *Procedia Technology*, 6, 126-135. doi:<https://doi.org/10.1016/j.protcy.2012.10.016>
- Rajakumar, B. R. (2020). Lion Algorithm and Its Applications. In M. Khosravy, N. Gupta, N. Patel, & T. Senjyu, *Frontier Applications of Nature Inspired Computation* (pp. 100-119). Springer Nature Singapore. doi:https://doi.org/10.1007/978-981-15-2133-1_5
- Ranjan, N. M., & Prasad, R. S. (n.d.). lion fuzzy neural network-based evolutionary model for text classification using context and sense based features. *Applied Soft Computing*. doi:<https://doi.org/10.1016/j.asoc.2018.07.016>
- RK, A., & UD, K. (2017). AFL-TOHIP: Adaptive fractional lion optimization to topology-hiding multi-path routing in mobile Ad hoc network. 727–732.
- Salim, B. W., Almufti, S. M., & Asaad, R. R. (2019). Review on elephant herding optimization algorithm performance in solving optimization problems. *International Journal of Engineering & Technology*, 7(4), 6109-6114. doi:10.14419/ijet.v7i4.23127
- Satish, C., P., V., & Praveen, D. (2017). Multi-objective-based adaptive dynamic directive operative fractional lion algorithm for data clustering. *International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)*. doi:10.1109/ICTUS.2017.8286066
- Y, L., Y, H., & M, Z. (2018). Short-term load forecasting for electric vehicle chargingstation based on niche immunity lion algorithm and convolutional neural network. *Energies*.
- Yazdani, M., & Jolai, F. (2016). Lion Optimization Algorithm (LOA): A nature-inspired. *Journal of Computational Design and Engineering*, 3(1). doi:<https://doi.org/10.1016/j.jcde.2015.06.003>
- Almufti, S. (2022). Lion algorithm: Overview, modifications and applications. *International Research Journal of Science*, 2(2), 176–186. <https://doi.org/10.5281/zenodo.6973555>

Chapter 4: Cuckoo Search Algorithm

1. Introduction

Traditional optimization techniques, such as gradient-based methods and exhaustive search, are effective for simple linear or convex problems but often struggle with the complexities of real-world scenarios. Gradient-based methods rely on the differentiability of the objective function, which may not always be available. They are also prone to becoming trapped in local optima, particularly in multi-modal landscapes. Similarly, exhaustive search methods, while guaranteed to find the global optimum, are computationally infeasible for large-scale problems due to the exponential growth of the solution space(Zou et al., 2015).

To address these limitations, metaheuristic algorithms have emerged as a robust alternative. These algorithms employ heuristic-based rules combined with stochastic elements to navigate complex search spaces efficiently (Almufti, 2019; Almufti et al., 2023).. Unlike traditional methods, metaheuristics do not guarantee the global optimum but aim to provide high-quality solutions within a reasonable timeframe. Their flexibility and adaptability make them well-suited for a wide range of optimization problems, including those with non-differentiable, discontinuous, or dynamic objective functions(S. Almufti, 2021).

Cuckoo Search Algorithm(CSA) Introduced by Xin-She Yang and Suash Deb in 2009 (Joshi et al., 2017) is inspired by the brood parasitism behaviour of certain cuckoo species. These birds lay their eggs in the nests of other species, relying on their hosts to incubate and raise the chicks. Some cuckoos enhance their reproductive success through mimicry, making their eggs visually similar to those of the host, thereby reducing the likelihood of rejection. In some cases, cuckoo chicks eliminate competition by pushing the host's eggs or chicks out of the nest, ensuring their own survival.

The biological behaviours of cuckoos are abstracted into an optimization process in CSA, where candidate solutions are modelled as nests. The quality of each solution is

represented as an egg's fitness, and the rejection of an egg by the host corresponds to the abandonment and replacement of poor solutions. The algorithm employs a Levy flight mechanism, a type of random walk characterized by step sizes drawn from a heavy-tailed probability distribution. This enables CSA to perform large exploratory jumps interspersed with smaller local refinements, achieving a balance between global exploration and local exploitation (Almufti et al., 2025).

The Cuckoo Search Algorithm has proven effective in addressing a wide range of optimization problems. Its simplicity and adaptability make it easy to implement, while its ability to escape local optima ensures robust performance even in complex, multi-modal landscapes. Additionally, CSA has been successfully modified and hybridized with other algorithms to enhance its performance for specific applications, such as engineering design, feature selection, and power flow optimization (Fister et al., 2014).

This chapter provides a comprehensive exploration of CSA, detailing its biological inspiration, mathematical formulation, and operational steps. A review of significant modifications is presented, highlighting hybrid models, adaptive mechanisms, and quantum-inspired approaches. Applications across various domains are discussed with detailed case studies and comparative analyses, demonstrating the algorithm's versatility and effectiveness. The chapter concludes with a discussion of current challenges and opportunities for future research, including the integration of CSA with advanced technologies like quantum computing and real-time optimization systems.

2. The Cuckoo Search Algorithm

Among the various metaheuristics, the Cuckoo Search Algorithm (CSA) stands out for its unique combination of simplicity, efficiency, and adaptability. Inspired by the brood parasitism behaviour of cuckoo birds, CSA uses a Levy flight mechanism to balance exploration and exploitation. Unlike many other algorithms, CSA does not require gradient information, making it suitable for non-differentiable and noisy optimization problems (Fister et al., 2014; Joshi et al., 2017; Yang & Suash Deb, 2009).

CSA's performance has been widely validated in domains such as engineering design, machine learning, and robotics, often outperforming other algorithms like GA and PSO in terms of convergence speed and solution quality. The inherent simplicity of CSA makes it easy to implement and adapt, further enhancing its appeal. Fig. 4-1 shows the flowchart of CSA

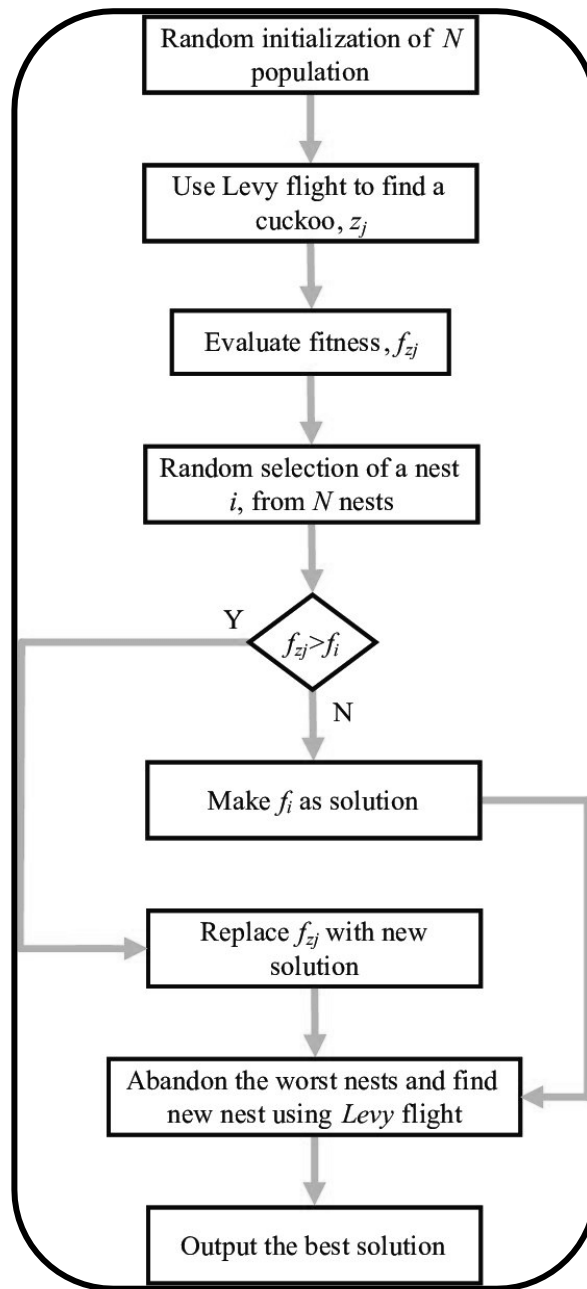


Fig. 4-1:CSA flowchart

2.1. Biological Inspiration

The Cuckoo Search Algorithm (CSA) draws inspiration from the brood parasitism behaviour of certain cuckoo bird species. These birds lay their eggs in the nests of other bird species, often removing the host's eggs to increase the likelihood that their offspring will survive. This parasitic behaviour allows cuckoos to avoid

investing time and energy in raising their young, relying instead on the host bird's efforts(Mareli & Twala, 2018).

Key aspects of this behaviour include(Soneji & Sanghvi, 2012):

- Selective Nesting: Female cuckoos select nests that maximize the chances of their eggs hatching undetected.
- Egg Mimicry: Some cuckoo species mimic the appearance of the host's eggs to reduce the probability of rejection.
- Chick Dominance: Cuckoo chicks often push host eggs or chicks out of the nest to eliminate competition.

These biological strategies are abstracted into the optimization framework of CSA, where:

- Eggs represent solutions.
- Nests correspond to candidate solutions in the search space.
- Host rejection of foreign eggs models the replacement of poor solutions.

2.2. Mathematical Framework

CSA operates in a continuous search space and employs Levy flights to update solutions. The following steps detail its mathematical formulation(Jati et al., 2012):

A. Initialization

The algorithm starts by initializing n candidate solutions (nests) randomly in the search space:

$$x_i^{(0)} \sim U(lower_bound, upper_bound)$$

where:

- $x_i^{(0)}$ is the i-th solution at iteration $t = 0$, U represents a uniform random distribution, lower_bound and upper_bound define the search space boundaries.

B. Levy Flight for Solution Update

New solutions are generated using Levy flights, which produce step lengths from a heavy-tailed distribution(Nguyen et al., 2016):

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \cdot Levy(\lambda)$$

where:

- $\alpha > 0$ is the step size scaling factor
- $Levy(\lambda)$ is a step length drawn from a Levy distribution.

The Levy distribution is defined as:

$$Levy(\lambda) \sim s^{-\lambda}, \text{ where } 1 < \lambda \leq 3,$$

and s is the step length. This allows CSA to make long-distance exploratory jumps and refine solutions locally.

In computational implementations, Levy flights are approximated as:

$$Levy = \frac{u}{|v|^{\frac{1}{\beta}}}$$

where:

- $u \sim N(0, \sigma_u^2)$ and $v \sim N(0, \sigma_v^2)$,

$$\sigma_u = \left[\frac{\Gamma(1+\beta) \cdot \sin(\frac{\pi\beta}{2})}{\left(\Gamma(\frac{1+\beta}{2}) \cdot \beta \cdot \frac{2^{\beta-1}}{2}\right)} \right]^{\frac{1}{\beta}},$$

- $\beta = 1.5$ is the typical Levy exponent.

C. Fitness Evaluation

Each solution is evaluated using a fitness function $f(x)$. The objective can be either to minimize or maximize $f(x)$, depending on the problem. For example(Sharma et al., 2021):

$$f(x) = \Sigma(x_i^2),$$

is a minimization problem, where the algorithm aims to find $x = 0$, which minimizes $f(x)$.

D. Abandonment and Replacement

A fraction P_a of the worst-performing solutions (nests) is abandoned and replaced with new random solutions:

$$x_i^{(t+1)} = \begin{cases} x_i^{(t+1)} & \text{if } r > P_a, \\ \text{random_solution} & \text{if } r \leq P_a \end{cases}$$

where $r \sim U(0, 1)$.

E. Selection

At the end of each iteration, the algorithm retains the best-performing solutions:

$$x_{best} = \operatorname{argmin} f(x),$$

for minimization problems.

Algorithmic Steps

The CSA algorithm can be summarized as follows:

1. Initialization: Generate n random solutions within the search space.
2. Fitness Evaluation: Compute the fitness $f(x)$ for all solutions.
3. Levy Flight Update: Update each solution using the Levy flight mechanism.
4. Replacement: Replace a fraction P_a of poor solutions with new random solutions.
5. Selection: Retain the best solutions for the next generation.
6. Termination: Stop when a maximum number of iterations or a satisfactory fitness level is achieved.

3. Modifications of the Cuckoo Search Algorithm

From the beginning CSA many modifications have been made on it, table 4-1 shows some CSA modifications (Almufti et al., 2025)

Table 4-1: popular modifications of CSA Algorithms

Modification	Purpose	Author	Year
Hybrid CSA-GA	Combines Genetic Algorithms with CSA to improve global search capabilities.	Yang & Deb	2009
Adaptive CSA	Dynamically adjusts parameters such as step size and abandonment rate for better convergence.	Zhang & Wu	2012
Discrete CSA	Adapts CSA for combinatorial and discrete optimization problems.	Kumar & Singh	2014
Chaotic CSA	Integrates chaotic maps for parameter initialization and control to enhance diversity.	Guo & Tang	2015
Multi-objective CSA	Extends CSA to solve multi-objective optimization problems using Pareto dominance.	Wang & Zheng	2016
CSA with Differential Evolution (CSA-DE)	Incorporates differential mutation and recombination strategies for improved balance between exploration and exploitation.	Sun & Zhang	2016
Parallel CSA	Implements parallel processing to reduce computation time for large-scale problems.	Almoosawi & Abdullah	2017

Quantum-Inspired CSA	Incorporates quantum computing principles to enhance exploration in high-dimensional spaces.	Zhao & Wang	2018
Memetic CSA	Combines CSA with local search heuristics for refining solutions locally.	Singh & Sharma	2018
Fuzzy CSA	Employs fuzzy logic to adaptively adjust algorithm parameters for uncertainty handling.	Chen & Li	2019
CSA with Levy Fraction Adjustment	Adjusts the Levy flight parameters dynamically based on solution quality and iteration progress.	Feng & Zhang	2019
CSA-PSO Hybrid	Integrates CSA with Particle Swarm Optimization to enhance local exploitation and global exploration.	Das & Roy	2020
Self-adaptive CSA	Uses self-adaptive parameter tuning to improve convergence in dynamic environments.	Liu	2020
Biogeography-based CSA	Incorporates biogeography-based concepts to improve diversity and solution quality.	He & Wang	2021
Enhanced CSA with Opposition-Based Learning	Applies opposition-based learning to improve the initial population and accelerate convergence.	Li	2021

4. CSA Applications

The Cuckoo Search Algorithm (CSA) is a powerful optimization algorithm inspired by the brood parasitism behaviour of cuckoo birds. It has been widely applied to solve complex optimization problems due to its simplicity, adaptability, and strong global search capabilities as show in table 4-2 (Almufti et al., 2025).

Table 4-2: Applications of the Cuckoo Search Algorithm

Domain	Application	Author	Year
Engineering	Structural optimization, such as truss design and weight minimization.	Smith et al.	2015
Machine Learning	Feature selection to improve model accuracy and reduce dimensions.	Johnson et al.	2017
Energy Systems	Optimal power flow to minimize losses and enhance grid efficiency.	Lee et al.	2018
Wireless Networks	Routing optimization to extend sensor network lifetime.	Singh et al.	2018
Robotics	Path planning for autonomous robots in dynamic environments.	Brown et al.	2019
Supply Chain	Inventory management and logistics optimization.	Lee & Kim	2019

Environmental Systems	Climate modeling and disaster prediction using optimization techniques.	Cooper et al.	2019
Transportation	Traffic flow optimization and vehicle routing.	Taylor et al.	2020
Cryptography	Secure key generation for encryption algorithms.	Wang et al.	2020
Biomedical Applications	Medical image segmentation and diagnostic tool enhancement.	Patel et al.	2020
Agriculture	Precision farming to optimize resource use and yield.	Zhao et al.	2020
Telecommunications	Spectrum allocation and network routing optimization.	Davis et al.	2020
Control Systems	PID controller tuning for stability and response optimization.	Adams et al.	2021
Economics	Portfolio optimization to maximize returns and minimize risk.	Gupta et al.	2021
Healthcare	Disease diagnosis and treatment planning.	Huang et al.	2021

5. Conclusion

The Cuckoo Search Algorithm (CSA) has emerged as a robust and versatile optimization tool, demonstrating remarkable adaptability and efficiency across diverse domains. Inspired by the brood parasitism behaviour of cuckoo birds, CSA leverages the Levy flight mechanism to balance exploration and exploitation, effectively addressing the challenges of non-linear, multi-modal, and high-dimensional optimization problems.

Through its inherent simplicity and flexibility, CSA has shown exceptional performance in solving problems such as structural optimization, feature selection, power flow optimization, and robotic path planning. Its ability to escape local optima and converge toward global solutions highlights its robustness against complex problem landscapes. Furthermore, the development of various modifications, including hybrid models, adaptive mechanisms, and domain-specific enhancements, has significantly expanded the algorithm's applicability and efficiency.

The comprehensive examples provided, from minimizing mathematical benchmarks to optimizing real-world systems like smart grids and wireless networks, underscore CSA's practical relevance. These applications illustrate the algorithm's potential to drive innovation in engineering, machine learning, energy systems, healthcare, and beyond.

Despite its success, CSA faces challenges, such as parameter sensitivity and computational demands for large-scale problems. Addressing these limitations through adaptive parameter tuning, parallel processing, and integration with emerging technologies like quantum computing offers exciting avenues for future research.

Additionally, exploring its synergy with other metaheuristic algorithms could unlock new levels of performance and applicability.

In conclusion, CSA stands as a cornerstone in the field of metaheuristics, offering a reliable and efficient framework for solving real-world optimization problems. Its continuous evolution through modifications and innovations ensures its relevance in addressing the growing complexity of modern systems. Researchers and practitioners alike can benefit from its adaptability and potential for driving impactful solutions in various fields.

6. References

- Acan, A., Altincay, H., Tekol, Y., & Unveren, A. (n.d.). A genetic algorithm with multiple crossover operators for optimal frequency assignment problem. The 2003 Congress on Evolutionary Computation, 2003. CEC '03., 256–263. <https://doi.org/10.1109/CEC.2003.1299583>
- Almufti, S. (2021). The novel Social Spider Optimization Algorithm: Overview, Modifications, and Applications. *ICONTECH INTERNATIONAL JOURNAL*, 5(2), 32–51. <https://doi.org/10.46291/icontechvol5iss2pp32-51>
- Almufti, S. M. (n.d.). U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem.
- Almufti, S. M., Marqas, R. B., Othman, P. S., & Sallow, A. B. (2021). Single-based and population-based metaheuristics for solving np-hard problems. *Iraqi Journal of Science*, 62(5), 1710–1720. <https://doi.org/10.24996/ijcs.2021.62.5.34>
- Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., & Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137, 106040. <https://doi.org/10.1016/j.cie.2019.106040>
- Evaluation of EHO, U-TACO and TS Metaheuristics algorithms in Solving TSP. (2020). *JOURNAL OF XI'AN UNIVERSITY OF ARCHITECTURE & TECHNOLOGY*, XII(IV). <https://doi.org/10.37896/jxat12.04/1062>
- Fister, I., Yang, X.-S., Fister, D., & Fister, I. (2014). Cuckoo Search: A Brief Literature Review (pp. 49–62). https://doi.org/10.1007/978-3-319-02141-6_3
- Gogna, A., & Tayal, A. (2013). Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*, 25(4), 503–526. <https://doi.org/10.1080/0952813X.2013.782347>
- Hedar, A.-R., & Fukushima, M. (2006). Derivative-Free Filter Simulated Annealing Method for Constrained Continuous Global Optimization. *Journal of Global Optimization*, 35(4), 521–549. <https://doi.org/10.1007/s10898-005-3693-z>
- Hwang, S.-F., & He, R.-S. (2006). A hybrid real-parameter genetic algorithm for function optimization. *Advanced Engineering Informatics*, 20(1), 7–21. <https://doi.org/10.1016/j.aei.2005.09.001>
- Jati, G. K., Manurung, H. M., & Suyanto. (2012). Discrete cuckoo search for traveling salesman problem. 2012 7th International Conference on Computing and Convergence Technology (IC CCT), 993–997.

- Joshi, A. S., Kulkarni, O., Kakandikar, G. M., & Nandedkar, V. M. (2017). Cuckoo Search Optimization- A Review. *Materials Today: Proceedings*, 4(8), 7262–7269. <https://doi.org/10.1016/j.matpr.2017.07.055>
- Kaveh, A., & Bakhshpoori, T. (2019). Metaheuristics: Outlines, MATLAB Codes and Examples. In *Metaheuristics: Outlines, MATLAB Codes and Examples*. Springer International Publishing. <https://doi.org/10.1007/978-3-030-04067-3>
- Kennedy, J., & Eberhart, R. (n.d.). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1942–1948. <https://doi.org/10.1109/ICNN.1995.488968>
- Liu, J. (2005). Novel orthogonal simulated annealing with fractional factorial analysis to solve global optimization problems. *Engineering Optimization*, 37(5), 499–519. <https://doi.org/10.1080/03052150500066646>
- M. Almufti, S. (2019). Historical survey on metaheuristics algorithms. *International Journal of Scientific World*, 7(1), 1. <https://doi.org/10.14419/ijsw.v7i1.29497>
- M. Almufti, S., Ahmad Shaban, A., Ismael Ali, R., & A. Dela Fuente, J. (2023). Overview of Metaheuristic Algorithms. *Polaris Global Journal of Scholarly Research and Trends*, 2(2), 10–32. <https://doi.org/10.58429/pgjsrt.v2n2a144>
- M. Almufti, S., Boya Marqas, R., & Ashqi Saeed, V. (2019). Taxonomy of bio-inspired optimization algorithms. *Journal of Advanced Computer Science & Technology*, 8(2), 23. <https://doi.org/10.14419/jacst.v8i2.29402>
- M. Almufti, S., Yahya Zebari, A., & Khalid Omer, H. (2019). A comparative study of particle swarm optimization and genetic algorithm. *Journal of Advanced Computer Science & Technology*, 8(2), 40. <https://doi.org/10.14419/jacst.v8i2.29401>
- Mareli, M., & Twala, B. (2018). An adaptive Cuckoo search algorithm for optimisation. *Applied Computing and Informatics*, 14(2), 107–115. <https://doi.org/10.1016/j.aci.2017.09.001>
- Nguyen, T. T., Vo, D. N., & Dinh, B. H. (2016). Cuckoo search algorithm for combined heat and power economic dispatch. *International Journal of Electrical Power & Energy Systems*, 81, 204–214. <https://doi.org/10.1016/j.ijepes.2016.02.026>
- Sharma, A., Sharma, A., Chowdary, V., Srivastava, A., & Joshi, P. (2021). Cuckoo Search Algorithm: A Review of Recent Variants and Engineering Applications (pp. 177–194). https://doi.org/10.1007/978-981-15-7571-6_8
- Shi, Y., & Eberhart, R. (n.d.). A modified particle swarm optimizer. 1998 IEEE International Conference on Evolutionary Computation Proceedings. *IEEE World Congress on Computational Intelligence* (Cat. No.98TH8360), 69–73. <https://doi.org/10.1109/ICEC.1998.699146>
- Soneji, H., & Sanghvi, R. C. (2012). Towards the improvement of Cuckoo search algorithm. 2012 World Congress on Information and Communication Technologies, 878–883. <https://doi.org/10.1109/WICT.2012.6409199>
- Xin-She Yang, & Suash Deb. (2010). Engineering optimisation by cuckoo search. *Int. J. Mathematical Modelling and Numerical Optimisation*, 1(4), 330–343.
- Yang, X.-S., & Suash Deb. (2009). Cuckoo Search via Levy flights. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), 210–214. <https://doi.org/10.1109/NABIC.2009.5393690>
- Yang, X.-She. (2010). *Nature-inspired metaheuristic algorithms*. Luniver Press.

- Zou, F., Wang, L., Hei, X., & Chen, D. (2015). Teaching–learning-based optimization with learning experience of other learners and its application. *Applied Soft Computing*, 37, 725–736. <https://doi.org/10.1016/j.asoc.2015.08.047>
- Almufti, S. M., Marqas, R. B., Asaad, R. R., & Shaban, A. A. (2025). Cuckoo search algorithm: overview, modifications, and applications. *International Journal of Scientific World*.

Chapter 5: Grey Wolf Optimizer Algorithm

1. Introduction

Grey Wolf Optimizer (GWO) developed by Mirjalili et al. in 2014 (Mirjalili et al., 2014). The GWO is a metaheuristic algorithm belongs to the third category (Nature-inspired). This algorithm is inspired by the hunting process found in Grey Wolves. This is unique as it follows the leadership hierarchy of the grey wolves. Grey wolves are well known for pack hunting and no other SI methods were proposed to follow this hierarchical hunting behaviour (Almufti, 2018). This chapter is an attempt to review the GWO algorithm and its proposed applications which help the researcher in future to apply it in other new and promising fields of application such as path planning and path following control of autonomous underwater vehicles (Marqas et al., 2020).

2. Grey Wolf Optimizer (GWO)

GWO is a swarm intelligence-based meta-heuristic algorithm that mimics the leadership hierarchy and hunting process of grey wolves in nature and mathematically models them (Al-Aboody et al., 2016; Panda et al., 2019). Grey wolves often prefer to live together. The population based on the importance is divided into four groups which are α , β , δ and ϵ , respectively. The alphas are the leaders of the group and are responsible for making decisions for hunting. The second level includes betas that help alphas in making decisions and other group activities. The omegas have the lowest level of hierarchy and are submissive of α , β and δ wolves. If a wolf is not a α , β or δ , then it is ϵ . The deltas follow α and β wolves but dominate the omega. The most important wolf groups are α , β and δ , respectively, that should guide omegas into areas with higher hunting probability (Panda et al., 2019; Mirjalili et al., 2015). Fig. 5-1 shows the steps of GWO Algorithm for solving optimizations problems.

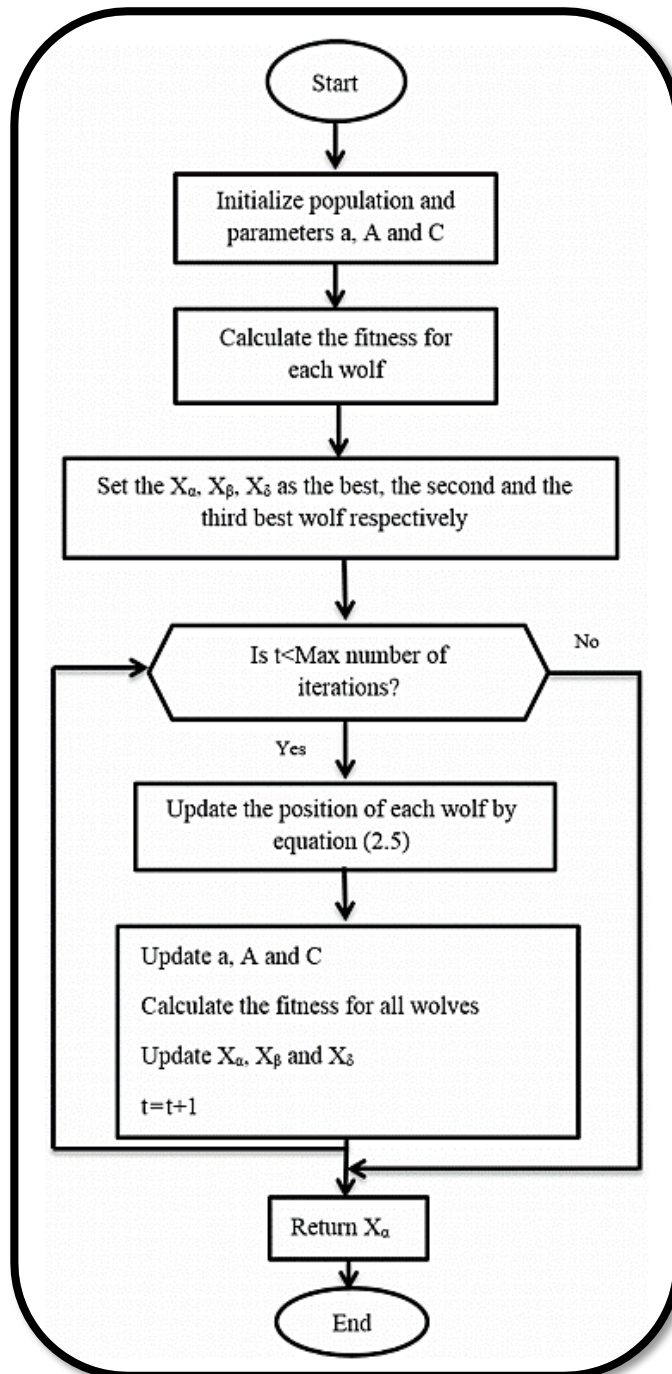


Fig. 5-1: GWO Flowchart

2.1. Inspiration of GWO

GWO is a swarm-based algorithm inspired by the social-intelligence of grey wolf leadership and hunting strategies. Generally grey wolf has four packs (Alpha, Beta, Delta, and Omega). In each pack of grey wolves, there is a common social hierarchy that dictates power and domination (see Fig. 3). In the Gray wolf hierarchy strategy, the most powerful wolf is alpha leads the entire pack in hunting's, migrations and feeding processes. In case of absence of alpha wolf from the pack, the strongest wolf of the beta pack wolfs takes the lead of the pack (Faris et al., 2017; Almufti, 2017a). The power and domination of the two other packs delta and omega are lower than alpha, and beta as can be seen in Fig. 5-2. This social intelligence is the main inspiration of the GWO algorithm. Another inspiration is the hunting approach of grey wolves. When hunting a prey, grey wolves follow a set of efficient steps: encircling and attacking. This allows them to hunt big preys (Almufti, 2017a; Negi et al., 2020).

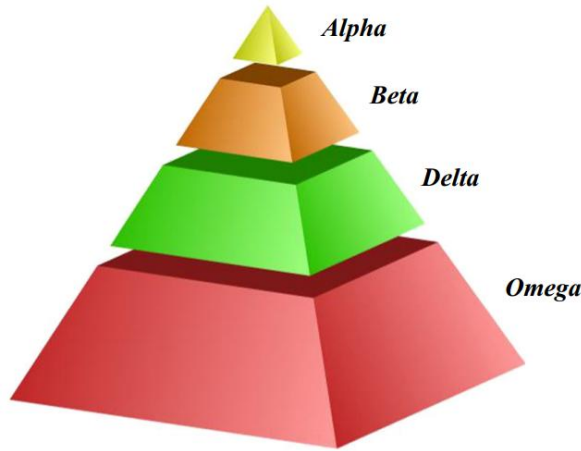


Fig. 5-2: Social hierarchy of grey wolves

2.2. mathematical Formulated of GWO

GWO algorithm divide into two phases includes (Mirjalili et al., 2015):

A. Encircling

As encircling phase, the first step in hunting process to chase and encircle. mathematically in this phase for a n-dimensional space GWO considers two wolfs (points) and it updates their location of the first one based on that of the second one. Based on Eq.1:

$$X(t + 1) = X(t) - A.D \quad (1)$$

where $X(t + 1)$ represents the next location of the wolf, $X(t)$ represent the current location, A is a coefficient matrix and D is a vector that depends on the location of the prey (X_p) and is calculated as show in eq.2.

$$D = |C \cdot X_p(t) - X(t)| \quad (2)$$

Where C can be calculated by eq. 3.

$$C = 2 \cdot r_2 \quad (3)$$

where r_2 is a randomly generated vector $\in [0,1]$. By using these two equations, a solution is able to relocate around another solution. Note that the equations use vectors, so this is applied to any number of dimensions. An example of possible positions of a grey wolf with respect to a prey is shown in Fig. 5-3.

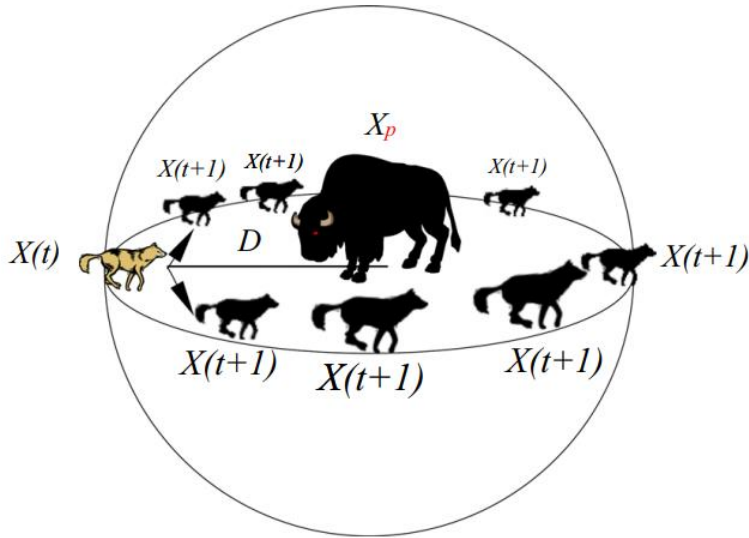


Fig. 5-3:Wolfs Position Updating

Note that the random components in the above equations simulate different step sizes and movement speeds of grey wolves. The equations to define their values are as follows:

$$A = 2a \cdot r_1 - a \quad (4)$$

where a is a vector where its values are linearly decreased from 2 to 0 during the course of run. r_1 is a randomly generated vector from the interval $[0,1]$. The equation to update the parameter a is as follows:

$$a = 2 - t \left(\frac{2}{T} \right) \quad (5)$$

where t shows the current iteration and T is the maximum number of iterations.

B. Attacking (Hunting) phase

With the equations presented above, a wolf can relocate to any points in a hypersphere around the prey. However, this is not enough to simulate the social intelligence of grey wolves. It was discussed above that social hierarchy plays a key role in hunt and the survival of a pack. To simulate social hierarchy, three best solutions are considered to be alpha, beta and delta. Although in nature there might be more than one wolf in each category, it is considered that there is only one solution belong to each class in GWO for the sake of simplicity.

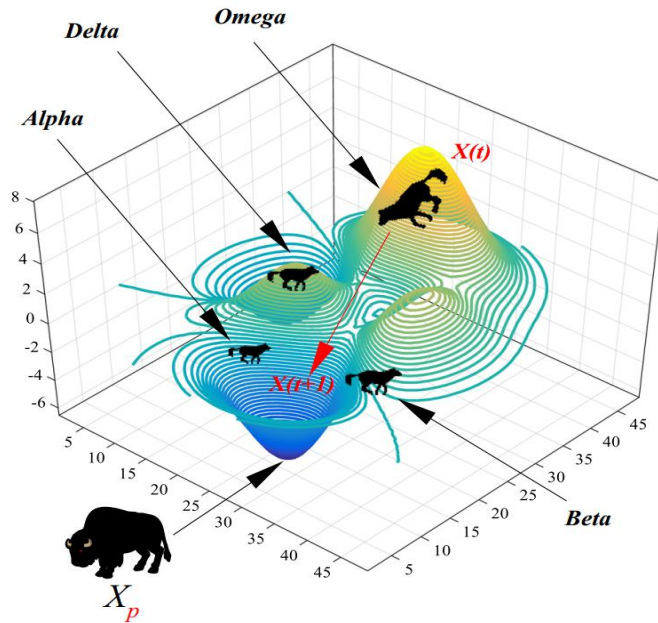


Fig. 5-4: alpha, beta, delta and omega in GWO

The concepts of alpha, beta, delta and omega are illustrated in Fig. 5-4. Note that the objective is to find the minimum in this search landscape. It may be seen in this figure that alpha is the closet solution to the minimum, followed by beta and delta. The rest of solutions are considered as omega wolves. There is just one omega wolf in Fig. 5, but there can be more. In GWO, it is assumed that alpha, beta and delta are always the three best solutions obtained so far. The global optimum of optimization problems is unknown, so it has been assumed that alpha, beta and delta have good idea of its location, which is reasonable because they are the best solutions in the entire population. Therefore, other wolves should be obliged to update their positions as follows:

$$X(t + 1) = \frac{1}{3}X_1 + \frac{1}{3}X_2 + \frac{1}{3}X_3 \quad (6)$$

where X_1 and X_2 and X_3 are calculated with Eq. 7.

$$\begin{aligned} X_1 &= X_\alpha(t) - A_1 \cdot D_\alpha \\ X_2 &= X_\beta(t) - A_2 \cdot D_\beta \\ X_3 &= X_\delta(t) - A_3 \cdot D_\delta \end{aligned} \quad (7)$$

Where D_α , D_β and D_δ can be calculated by Eq. 8.

$$\begin{aligned} D_\alpha &= |C_1 \cdot X_\alpha - X| \\ D_\beta &= |C_2 \cdot X_\beta - X| \\ D_\delta &= |C_3 \cdot X_\delta - X| \end{aligned} \quad (8)$$

3. Modifications of GWO

Generally, all the metaheuristics algorithms after its first appearance undergoes many modifications and improvements, so that it can be used to solve various problems (Almufti, 2021b; Wen et al., 2015).

After the appearance of Grey Wolf Optimizer (GWO) algorithm in 2014, To meet the demands of the real-world problems many modifications have been applied to the original GWO to improve the performances of the proposed algorithm. The modifications were in two various ways:

- **Unhybridized GWO:** This form includes the original or the standard GWO as proposed by Mirjalili (Mirjalili et al., 2014) and all the varied forms of GWO which have been modified with different parameters and approaches to the Original GWO along with the different variants (Almufti, 2017a).
- **Hybridized GWO:** Hybridization means a combination of two or more algorithms to utilize the benefits of the characteristics of each algorithm in the best possible way. The hybridized algorithm showed competitive results in terms of convergence rate, quality results, efficiency, and stability than the individual algorithms in most cases (Almufti, 2017a).

In this section, some of the modifications and improvements of the GWO algorithm are listed and arranged according to the development year (Almufti et al., 2021c), as shown in Table 5-1.

Table 5-1: GWO based algorithms

#	Abbr.	Name	Author	Type	Year	Ref.
1.	IGWO	Improved GWO	Wen et al.	Unhybridized	2015	(Wen et al.,2015)
2.	CEGWO	Complex-valued encoding GWO	Luo et al.	Unhybridized	2015	(Luo et al., 2015)
3.	BGWO	Binary GWO	Emary et al.	Unhybridized	2016	(Emary et al., 2015)
4.	MGWO	Modified GWO	Mittal et al.	Unhybridized	2016	(Mittal et al., 2016)
5.	MDGWO	Modified discrete GWO	Li et al.	Unhybridized	2016	(Li et al., 2016)
6.	MOGWO	Multi-objective GWO	Mirjalili et al.	Unhybridized	2016	(Marijalili et al., 2016)
7.	EGWO	Enhanced GWO	Joshi and Arora	Unhybridized	2017	(Joshi et al., 2017)
8.	GWO-PSO	GWO- particle swarm optimizer	Singh and Singh	Hybridized	2017	(Singh et al., 2017a)
9.	GWO-ACO	GWO-Ant Colony Optimization	Ab Rashid	Hybridized	2017	(Ab,2017)
10	GWO-GA	GWO- Genetic Algorithm	Tawhid and Ali	Hybridized	2017	(Tawhid et al., 2017)
11	GWO-SCA	GWO- Sine Cosine Algorithm	Singh and Singh	Hybridized	2017	(Singh et al., 2017b)
12	CGWO	Chaotic GWO	Kohli and Arora	Unhybridized	2018	(Kohli et al., 2018)
13	EEGWO	Exploration-enhanced GWO	Long et al.	Unhybridized	2018	(Long et al., 2018)
14	INGWO	Intelligent GWO	Liu et al.	Unhybridized	2018	(Liu et al., 2018)
15	VWGWO	variable weights GWO	Gao and Zhao	Unhybridized	2019	(Gao et al., 2019)
16	RLGWO	Refraction learning-GWO	Long et al.	Unhybridized	2019	(Long et al., 2019)

4. GWO Applications

Over year, Grey Wolf Optimizer (GWO) algorithm and its modifications showed high-performance in solving different real-world problems and it has been used for solving unconstrained, constrained, multi-objective and NP-Hard problems in fields of engineering, medical, environment (Almufti et al, 2021c), as summarized in Table 5-2.

Table 5-2: summarize GWO applications in different field

#	Application	Discussion	Ref.
1.	Feature selection	Feature selection is one of the important processes in machine learning and data mining. The goal of feature selection is to reduce the number of features, select the most representative ones and to eliminate redundant, noisy and irrelevant features. The problem of searching for best set of features is considered as complex and difficult problem due to the extremely large search space when the number of features is large.	(Negi et al., 2020) (Mittal et al., 2016) (Eary et al., 2015) (Vosooghifard et al., 2015) (Mejahed et al., 2016) (Yamany et al., 2016)
2.	Clustering applications	Clustering is a common machine learning and data mining task where the goal is to divide data instances into several groups that have similar characteristics in some sense.	(Kumar et al., 2017) (Zhang et al., 2015) (Yang et al., 2015)
3.	Training neural networks	Artificial neural networks (ANNs) are information processing models inspired by the biological nervous systems. ANNs are widely applied in research and practice due to their high capability for capturing nonlinearity and dynamicity. However, the performance of ANNs is highly affected by their structure and connection weights. Traditionally, the efficiency of any new metaheuristic algorithm is investigated in optimizing the connection weights neural networks shortly after its release.	(Mirjalili, 2015b) (Mosavi et al., 2016) (Mohamed et al., 2015) (Almufti,2021)
4.	Design and tuning controllers	In control engineering, we have noticed an increased number of publications that investigate the application of GWO in tuning the parameters of controllers such as integral (I), proportional-integral (PI), proportional-integral-derivative (PID).	(Li et al., 2015) (Yadav et al., 2016) (Das et al., 2015)
5.	Knapsack Problem	The knapsack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.	(Yassien et al., 2017)

6. Power dispatch problems	The economic load dispatch (ELD) is a class of non-convex and highly nonlinear constrained optimization problem concern with finding an optimal load dispatch to operate and plan the current resources. This problem is a kind of optimization problem in which its complexity is increased based on the number of system units to be planned. It is concern with distributing the required electricity among the generating units in optimum way in order to minimize the fuel consumptions of each unit in accordance with a power balance equality constraints and power output inequality constraints.	(Wong et al., 2014) (Song et al., 2014) (jayabarathi et al., 2016)
7. Robotics and path planning	Path-planning is an important primitive for autonomous mobile robots that lets robots find the shortest – or otherwise optimal – path between two points. Otherwise, optimal paths could be paths that minimize the amount of turning, the amount of braking or whatever a specific application requires.	(Zhang et al., 2015) (Tsai et al., 2016) (Korayem et al., 2015)
8. Wireless sensor network	Generally, GWO had a high-performance in Wireless sensor network applications. It has been used in Tackling the coverage problem in WSN, routing and localization problem in WSNs	(Dao, 2016) (Al-Aboody et al., 2016) (Fouad et al., 2015)
9. Scheduling	One common scheduling problem is the job shop, in which multiple jobs are processed on several machines. Each job consists of a sequence of tasks, which must be performed in a given order, and each task must be processed on a specific machine. GWO has been used to solve scheduling problem	(Lu et al., 2015) (Lu et al., 2016)
10 Surface Wave Dispersion Curve	“Surface Wave Dispersion Curve Inversion” scheme based on GWO. The proposed method has been verified on environments both with and without noise and proved to be efficient.	(Song et al., 2015)
11 Biomedical Research	GWO can also be employed for modelling modular granular neural networks (MGNN). The proposed design performed optimal granulation of data for human recognition	(Sanchez et al., 2017) (Parsian et al., 2017)
12 NP-hard problem	Generally, GWO had a high-performance in solving NP-Hard problems	(Korayem et al., 2015)

5. Conclusion

Grey Wolf Optimizer (GWO) algorithm is a Swarm-based metaheuristic algorithm proposed in 2014 by Mirjalili et al. After its appearance, many modifications have been proposed on it and it has been adapted to solve various problem in different fields. This chapter firstly addressed this overviewed the original GWO algorithm and then it presented some of its modifications were presented in detail, finally some of its application considering parameter tuning, different approaches for feature selection and classification, and then hybridized forms were discussed. The applications in different

fields were discussed including engineering medical, power dispatch, reliability optimization etc.

6. REFERENCES

- Ab Rashid MFF (2017) A hybrid Ant-Wolf Algorithm to optimize assembly sequence planning problem. *Assembly Autom* 37(2):238–248
- Al-Aboody NA, Al-Raweshidy HS (2016) Grey wolf optimization-based energy-efficient routing protocol for heterogeneous wireless sensor networks. In: 2016 4th international symposium on computational and business intelligence (ISCBI). IEEE, pp 101–107
- Almufti S., (2018), U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem, Hdl.handle.net, 2018. [Online].
- Almufti, S. (2017a). Using Swarm Intelligence for solving NPHard Problems. *Academic Journal Of Nawroz University*, 6(3), 46-50. doi: 10.25007/ajnu.v6n3a78
- Almufti, S. (2019a). Historical survey on metaheuristics algorithms. *International Journal Of Scientific World*, 7(1), 1. doi: 10.14419/ijsw.v7i1.29497
- Almufti, S. (2021b). The novel Social Spider Optimization Algorithm: Overview, Modifications, and Applications. *ICONTECH INTERNATIONAL JOURNAL*, 5(2), 32-51. doi: 10.46291/icontechvol5iss2pp32-51
- Almufti, S., Marqas, R., Othman, P., & Sallow, A. (2021a). Single-based and Population-Based Metaheuristics Algorithms Performances in Solving NP-hard Problems. *Iraqi Journal Of Science*. doi: 10.24996/10.24996/ij.s.2021.62.5.34
- Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. *Adv Eng Softw* 69:46–61
- Amirsadri, S., Mousavirad, S., & Ebrahimpour-Komleh, H. (2017). A Levy flight-based grey wolf optimizer combined with back-propagation algorithm for neural network training. *Neural Computing And Applications*, 30(12), 3707-3720. doi: 10.1007/s00521-017-2952-5
- Asaad, R., & Abdunabi, N. (2018). Using Local Searches Algorithms with Ant Colony Optimization for the Solution of TSP Problems. *Academic Journal Of Nawroz University*, 7(3), 1-6. doi: 10.25007/ajnu.v7n3a193
- Balamurugan, R., Natarajan, A., & Premalatha, K. (2015). Stellar-Mass Black Hole Optimization for Biclustering Microarray Gene Expression Data. *Applied Artificial Intelligence*, 29(4), 353-381. doi: 10.1080/08839514.2015.1016391
- Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ (2009) A survey on metaheuristics for stochastic combinatorial optimization. *Nat Comput* 8(2):239–287.
- Dao TK (2016) Enhanced diversity herds grey wolf optimizer for optimal area coverage in wireless sensor networks. In: Genetic and evolutionary computing: proceedings of the tenth international conference on genetic and evolutionary computing, November 7–9, 2016 Fuzhou City, Fujian Province, China, vol 536. Springer, p 174
- Das KR, Das D, Das J (2015) Optimal tuning of pid controller using gwo algorithm for speed control in dc motor. In: 2015 international conference on soft computing techniques and implementations (ICSCTI). IEEE, pp 108–112.
- Eary E, Yamany W, Hassanien AE, Snasel V (2015) Multiobjective gray-wolf optimization for attribute reduction. *Procedia Comput Sci* 65:623–632

- Emary E, Zawbaa HM, Hassanien AE (2016) Binary grey wolf optimization approaches for feature selection. *Neurocomputing* 172:371–381
- Faris, H., Aljarah, I., Al-Betar, M., & Mirjalili, S. (2017). Grey wolf optimizer: a review of recent variants and applications. *Neural Computing And Applications*, 30(2), 413-435. doi: 10.1007/s00521-017-3272-5
- Fouad MM, Hafez AI, Hassanien AE, Snasel V (2015) Grey wolves optimizer-based localization approach in WSNs. In: 2015 11th international computer engineering conference (ICENCO). IEEE, pp 256–260.
- Gao ZM, Zhao J (2019) An improved grey wolf optimization algorithm with variable weights. *Comput Intell Neurosci*. <https://doi.org/10.1155/2019/2981282>.
- Glover F.,(1986), Future paths for integer programming and links to artificial intelligence, *Computers & Operations Research*, vol. 13, no. 5, pp. 533-549. Available: 10.1016/0305-0548(86)90048-1. [https://doi.org/10.1016/0305-0548\(86\)90048-1](https://doi.org/10.1016/0305-0548(86)90048-1).
- Harifi, S., Mohammadzadeh, J., Khalilian, M., & Ebrahimnejad, S. (2020). Giza Pyramids Construction: an ancient-inspired metaheuristic algorithm for optimization. *Evolutionary Intelligence*. doi: 10.1007/s12065-020-00451-3
- Ihsan, R., Almufti, S., Ormani, B., Asaad, R., & Marqas, R. (2021). A Survey on Cat Swarm Optimization Algorithm. *Asian Journal Of Research In Computer Science*, 22-32. doi: 10.9734/ajrcos/2021/v10i230237.
- Jayabarathi T, Raghunathan T, Adarsh BR, Suganthan PN (2016) Economic dispatch using hybrid grey wolf optimizer. *Energy* 111:630–641
- Joshi H, Arora S (2017) Enhanced grey wolf optimisation algorithm for constrained optimisation problems. *Int J Swarm Intell* 3(2–3):126–151
- Kohli M, Arora S (2018) Chaotic grey wolf optimization algorithm for constrained optimization problems. *J Comput Design Eng* 5(4):458–472
- Korayem L, Khorsid M, Kassem SS (2015) Using grey wolf algorithm to solve the capacitated vehicle routing problem. In: *IOP conference series: materials science and engineering*, vol 83. IOP Publishing, p 012014
- Korayem L, Khorsid M, Kassem SS (2015) Using Grey Wolf algorithm to solve the capacitated vehicle routing problem. In: *IOP conference series: materials science and engineering*, vol 83, no 1. IOP Publishing, p 012014.
- Kumar V, Chhabra JK, Kumar D (2017) Grey wolf algorithmbased clustering technique. *J Intell Syst* 26(1):153–168
- Li L, Sun L, Kang W, Guo J, Han C, Li S (2016) Fuzzy multilevel image thresholding based on modified discrete grey wolf optimizer and local information aggregation. *IEEE Access* 4:6438–6450
- Li SX, Wang JS (2015) Dynamic modeling of steam condenser and design of pi controller based on grey wolf optimizer. *Math Probl Eng* 2015:9
- Liu H, Hua G, Yin H, Xu Y (2018) An intelligent grey wolf optimizer algorithm for distributed compressed sensing. *Comput Intell Neurosci*. <https://doi.org/10.1155/2018/1723191>
- Long W, Jiao J, Liang X, Tang M (2018) An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization. *Eng Appl Artif Intell* 68:63–80
- Long W, Wu T, Cai S, Liang X, Jiao J, Xu M (2019) A novel grey wolf optimizer algorithm with refraction learning. *IEEE Access* 7:57805–57819

- Lu C, Gao L, Li X, Xiao S (2017) A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry. *Eng Appl Artif Intell* 57:61–79
- Lu C, Xiao S, Li X, Gao L (2016) An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production. *Adv Eng Softw* 99:161–176
- Luo Q, Zhang S, Li Z, Zhou Y (2015) A novel complex-valued encoding grey wolf optimization algorithm. *Algorithms* 9(1):4
- Marashdih, A.W., Zaaba, Z.F., Almufti, S.M. (2018). The Problems and Challenges of Infeasible Paths in Static Analysis. *Int. J. Eng. Technol.* 2018,7, 412–417.
- Marqas, R., Almufti, S., & Asaad, R. (2019). Comparative study between elephant herding optimization (EHO) and U-turning ant colony optimization (U-TACO) in solving symmetric traveling salesman problem (STSP). *Journal Of Advanced Computer Science & Technology*, 8(2), 32. doi: 10.14419/jacst.v8i2.29403.
- Marqas, R., Almufti, S., Othman, P., & Abdulrahman, C. (2020). Evaluation of EHO, U-TACO and TS Metaheuristics algorithms in Solving TSP. *JOURNAL OF XI'an UNIVERSITY OF ARCHITECTURE & TECHNOLOGY*, XII(IV). doi: 10.37896/jxat12.04/1062.
- Medjahed SA, Ait ST, Benyettou A, Ouali M (2016) Gray wolf optimizer for hyperspectral band selection. *Appl Soft Comput* 40:178–186
- Mirjalili S (2015) How effective is the Grey wolf optimizer in training multi-layer perceptrons. *Appl Intell* 43(1):150–161
- Mirjalili S (2015b) How effective is the grey wolf optimizer in training multi-layer perceptrons. *Appl Intell* 43(1):150–161
- Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Comput Appl* 27(2):495–513
- Mittal N, Singh U, Sohi BS (2016) Modified grey wolf optimizer for global engineering optimization. *Appl Comput Intell Soft Comput* 2016:8
- Mohamed AAA, El-Gaafary AAM, Mohamed YS, Hemeida AM (2015) Design static var compensator controller using artificial neural network optimized by modify grey wolf optimization. In: 2015 international joint conference on neural networks (IJCNN). IEEE, pp 1–7
- Mosavi MR, Khishe M, Ghamgosar A (2016) Classification of sonar data set using neural network trained by gray wolf optimization. *Neural Netw World* 26(4):393
- Negi G., Kumar A., Pant S. and Ram M., (2020), GWO: a review and applications, *International Journal of System Assurance Engineering and Management*, vol. 12, no. 1, pp. 1-8,. Available: 10.1007/s13198-020-00995-8 [Accessed 2 August 2021].
- Panda, M., & Das, B. (2019). Grey Wolf Optimizer and Its Applications: A Survey. *Lecture Notes In Electrical Engineering*, 179-194. doi: 10.1007/978-981-13-7091-5_17
- Parsian A, Ramezani M, Ghadimi N (2017) A hybrid neural network-Gray Wolf optimization algorithm for melanoma detection. *Biomed Res* 28(8)
- Saeed, V., Marqas, R., & Almufti, S. (2019). Taxonomy of bio-inspired optimization algorithms. *Journal Of Advanced Computer Science & Technology*, 8(2), 23. doi: 10.14419/jacst.v8i2.29402.
- Salim B., Almufti, S., R. Asaad, (2018). Review on Elephant Herding Optimization Algorithm Performance in Solving Optimization Problems. *International Journal of Engineering & Technology* 7 : 6109-6114

- Sanchez D, Melin P, Castillo O (2017) A Grey Wolf optimizer for modular granular neural networks for human recognition. *Comput Intell Neurosci*
- Shaban, A., & Almufti, S. (2018). U-Turning Ant Colony Algorithm for Solving Symmetric Traveling Salesman Problem. *Academic Journal Of Nawroz University*, 7(4), 45-49. doi: 10.25007/ajnu.v6n4a270
- Singh N, Singh SB (2017a) Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. *J Appl Math*. <https://doi.org/10.1155/2018/1723191>
- Singh N, Singh SB (2017b) A novel hybrid GWO-SCA approach for optimization problems. *Eng Sci Technol Int J* 20(6):1586–1601
- Song HM, Sulaiman MH, Mohamed MR (2014) An application of grey wolf optimizer for solving combined economic emission dispatch problems. *Int Rev Model Simul (IREMOS)* 7(5):838–844
- Song X, Tang L, Zhao S, Zhang X, Li L, Huang J, Cai W (2015) Grey Wolf optimizer for parameter estimation in surface waves. *Soil Dyn Earthq Eng* 75:147–157.
- Tawhid MA, Ali AF (2017) A Hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function. *Memetic Comput* 9(4):347–359
- Tsai PW, Dao TK, et al (2016) Robot path planning optimization based on multiobjective grey wolf optimizer. In: *International conference on genetic and evolutionary computing*. Springer, pp 166–173
- Vosooghifard M, Ebrahimpour H (2015) Applying grey wolf optimizer-based decision tree classifier for cancer classification on gene expression data. In: *2015 5th international conference on computer and knowledge engineering (ICCCKE)*. IEEE, pp 147–151
- Wen L, Dongquan Z, Songjin XU (2015) Improved Grey Wolf Optimization algorithm for constrained optimization problem. *J Comput Appl* 35(9):2590–2595
- Wong LI, Sulaiman MH, Mohamed MR, Hong MS (2014) Grey wolf optimizer for solving economic dispatch problems. In: *2014 IEEE international conference on power and energy (PECon)*. IEEE, pp 150–154
- Yadav S, Verma SK, Nagar SK (2016) Optimized pid controller for magnetic levitation system. *IFAC-ChaptersOnLine* 49(1):778–782
- Yahya Zebari, A., Almufti, S., & Khalid Omer, H. (2019). A comparative study of particle swarm optimization and genetic algorithm. *Journal Of Advanced Computer Science & Technology*, 8(2), 40. doi: 10.14419/jacst.v8i2.29401
- Yahya Zebari, A., M. Almufti, S., & Mohammed Abdulrahman, C. (2020). Bat algorithm (BA): review, applications and modifications. *International Journal Of Scientific World*, 8(1), 1. doi: 10.14419/ijsw.v8i1.30120
- Yamany W, Emary E, Hassanien AE (2016) New rough set attribute reduction algorithm based on grey wolf optimization. In: *The 1st international conference on advanced intelligent system and informatics (AISi2015)*, November 28–30, 2015, Beni Suef, Egypt. Springer, pp 241–251
- Yang H, Liu J (2015) A hybrid clustering algorithm based on grey wolf optimizer and k-means algorithm. *J Jiangxi Univ Sci Technol* 5:015
- Yassien E, Masadeh R, Alzaqebah A, Shaheen (2017) A Grey Wolf optimization applied to the 0/1 knapsack problem. *Int J Comput Appl* (0975–8887) 169(5)

- Zhang S, Zhou Y (2015) Grey wolf optimizer based on Powell local optimization method for clustering analysis. *Discret Dyn Nat Soc* 2015:17
- Almufti, S. M., Marqas, H. B., & Asaad, R. B. (2021c). Grey wolf optimizer: Overview, modifications and applications. *International Research Journal of Science*, 1(1), 44–56. <https://doi.org/10.5281/zenodo.5195644>

Chapter 6: Vibrating Particles System Algorithm:

1. Introduction

Vibration is a mechanical phenomenon that causes oscillations around an equilibrium state. It is one of well-known Single-Objective Constrained Optimization Problems (SOCOP) in the engineering fields. Vibrations are classified into two types: (1) free vibration and (2) forced vibration (Celik & Kutucu, 2018). This chapter present Vibrating Particles System (VPS) algorithm, which is a Physics-Based algorithm belongs to Physics-inspired category of Metaheuristics algorithms, it is developed by Kaveh et al. in 2017 (Kaveh & Ghazaan, 2017). The VPS algorithm is inspired by the free vibration of an under-damping objects. This chapter is an attempt to review the VPS algorithm and its proposed applications which help the researcher in future to apply it in other new and promising fields of application such as path planning and path following control of autonomous underwater vehicles.

2. VIBRATING PARTICLES SYSTEM (VPS)

The Vibrating Particles System (VPS) approach, developed by Kaveh and Ilchi Ghazaan in 2017, is an evolutionary metaheuristic search strategy that stimulates the free vibration of single degree of freedom systems with viscous dampening. VPS has been used to address a variety of structural optimization problems, with the results demonstrating its feasibility in terms of convergence and accuracy (Kaveh & Bakhshpoor, Metaheuristics: Outlines, MATLAB Codes and Examples, 2019).

VPS, like other population-based metaheuristics, begins with a random collection of starting solutions and treats them as free vibrated single degree of freedom systems with vibration. When dampening conditions are applied to a free vibrating system, it oscillates and returns to its equilibrium point with a specified formulation. This is simply proven using differential equations. As the optimization process progresses, VPS enhances the particle quality on a regular basis by oscillating them forward towards the equilibrium

position using a mix of randomization and exploitation of the data gained (Almufti et al.,2022a).

Consider that each particle's equilibrium position is made up of three positions: the best/highest position (HP), a good particle (GP), and a poor particle (BP). Thus, the essence of VPS is founded on three key concepts:

- self-adaptation: the particle moves toward HP.
- Collaborations: GP and BP, which are chosen from among the particles, can influence the new particle location.
- Competitions: GP's influence surpasses that of BP.

It should be noted that VPS uses a memory based on the harmony search approach to adjust the location of particles departing the search field (Ou, Zeng, Chang, & Lin, 2020).

2.1. VPS Formulation

The formula employed to characterize the free vibration of an under-damped single degree of freedom (SDOF) system as follows in Eq(3).

$$X(t) = \rho e^{-\varepsilon \omega_n t} \sin(\omega_D t + \theta) \quad (3)$$

where ρ and θ are constants that are typically derived from the vibration's beginning circumstances, ω_n is the vibration's natural circular frequency. ω_D and ε are respectively, the damped natural frequency and the damping ratio that are shown in Fig 6.1, and can be calculated by Eq(4) and Eq(5).

$$\omega_D = \omega_n \sqrt{1 - \varepsilon^2} \quad (4)$$

$$\varepsilon = \frac{c}{2m\omega_n} \quad (5)$$

All particles' initial positions in the search space are generated at random by Eq(6):

$$X_j^i = X_{min} + rand * (X_{max} - X_{min}) \quad (6)$$

Where X_{min} and X_{max} are the minimum and the maximum allowable variables

vectors. And rand is a random number uniformly distributed in the range of [0, 1].

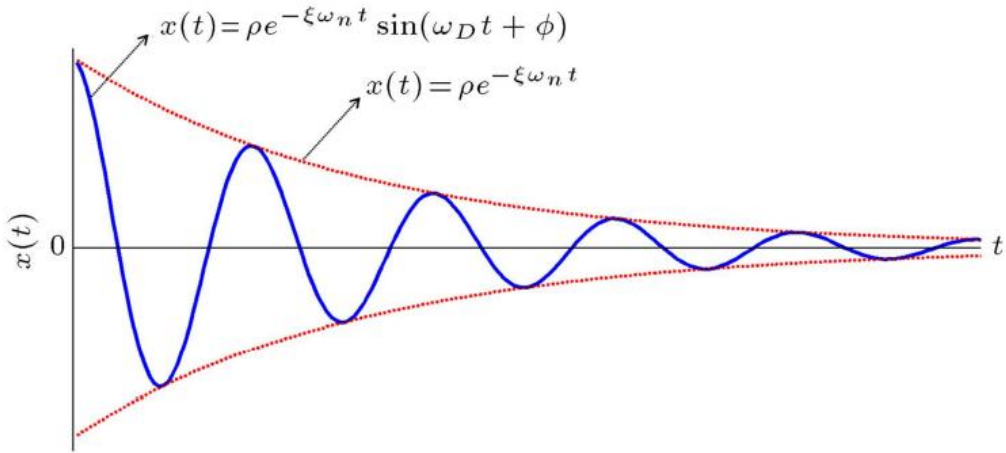


Fig. 6-1: Vibrating motion of under-damped system.

As is evident, an under-damping SDOFs free vibrate and slowly comes into a its equilibrium point. And this vibrating characteristics is the idea which the VPS is inspires.

similar to other metaheuristics algorithms, VPS begins with a set of intrant solutions that are randomly produced inside the search space. nVP stands for the number of vibrated-particle. The appropriate objective-functions (*Fit*) and its penalized-functions (*PFit*) are generated once the items are evaluated (Kaveh & Bakhshpoor, Metaheuristics: Outlines, MATLAB Codes and Examples, 2019).

VPS updates HP, GP, and BP with various weights (ω_1, ω_2 , and ω_3) for each particle. After the population is sorted in ascending order by the values of the penalized objective function. For each particle, GP and BP are randomly selected from the first and second halves of the population, with the exception of itself. The damping level has a significant impact on vibration. The amplitude of a free damped vibration decreases as the damping level increases. To mimic this behavior in the VPS, the following descending function (D) relative to number of repetitions is calculated by Eq(7):

$$D = \left(\frac{NITs}{maxNITs} \right)^\alpha \quad (7)$$

$$maxNITs = \frac{maxNFES}{nVP} \quad (8)$$

where NITs is the algorithm's current iteration number, maxNITs is the number of algorithm iterations selected as the halting criterion, maxNFEs is the number of function evaluations used as the stopping criterion, and α is a constant. Picking a value of 0.05 is advised (Kaveh & Ghazaan, Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints, 2017).

The particles are updated by the equation Eq(9,10,11), often known as the free vibration formula, in accordance with the aforementioned ideas:

$$\begin{aligned} newVP_i = & \omega_1(D * A * rand_1 + HP) \\ & + \omega_2(D * A * rand_2 + GP_i) \\ & + \omega_3(D * A * rand_3 + BP_i) \end{aligned} \quad (9)$$

$$A = \omega_1(HP - VP_i) + \omega_2(GP_i - VP_i) + \omega_3(BP_i - VP_i) \quad (10)$$

$$\omega_1 + \omega_2 + \omega_3 = 1 \quad (11)$$

Considering the VP_i as the current locations and $newVP_i$ as the updated locations of the i th particle's. The qualified significance of the i th particle's good particle, bad particle, and algorithm's best-yet particle are compared using different constant weights ω_1, ω_2 , and ω_3 , and $rand$ represents a random value uniformly generated in $[0 - 1]$ range.

In Eq(9 and 10), A and D effects on algorithm process similar to the effect of ρ and $e^{-\varepsilon\omega_n t}$ in Eq(3). The value of $\sin(\omega_D t + \theta)$ is measured as unity. A parameter p between $(0, 1)$ is initialized, and it specifies the influence of BP should/shouldn't considers when position are updating. p and $rand$ are compared as shown in Eq(12):

$$\text{If } P < rand \rightarrow \omega_3 = 0, \omega_2 = 1 - \omega_1 \quad (12)$$

Self-adaptation, collaboration, and competitiveness are three fundamental ideas that VPS takes into consideration (Kaveh & Ghazaan, Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints, 2017).

- Self-adaptation happens as a particle moves in the direction of HP.
- Cooperation between particles is offered in VPS, where every particle has the capacity to influence the other's new location.

- competition :The p parameter causes competition to arise because the effect of the GP (good particle) is larger than that of the BP (bad particle)

The VPS uses a harmony search-based handling method to deal with a particle that has gone outside the boundaries of the variables. The maximum vibrating particle's nVPs number, together with its related objective function (FitM) and penalized (PFit-M) values, are kept in a vibrating-particle memory in this research (VP-M). Vibrating particle memory was employed to store numbers as equal to nVP in order to accomplish this. Memory consideration and use in diverse approaches can increase metaheuristics efficiency without increasing processing costs. It should be noted that VPS utilized memory just to renew particles that had left the search region. This technique allows any member of the solution set that deviates from the bounds to be regenerated from the VP-M and is determined by Eq (13).

$$VP(i, j) = \begin{cases} w.p. \text{ } vpmcr \Rightarrow & \text{select a new value for a variable from VP} \\ & w.p. (1 - par) \text{ do nothing,} \\ & w.p. \text{ } par \text{ choose a neighboring value.} \\ w.p. (1 - vpmcr) \Rightarrow & \text{Select a new value randomly} \end{cases} \quad (13)$$

where "w.p." stands for "with the probability," and $VP(i, j)$ is the j th element of the i th vibrated particle, $vpmcr$ is the vibrating-particles memory with a rate within $[0,1]$ and determines the possibility of selecting a value from the historic values saved in $VP - M$, and $(1 - vpmcr)$ determines the probability randomly selecting a values in the possible range. After selecting a value from $VP - M$, the pitch-adjusting process begins. The value $(1 - par)$ specifies the neglecting rate, and par specifies the rate of selecting a value from the particles bordering the best vibrating particle or those preserved in memory. Random generating step size ($\pm bw * rand$) can be used for continuous search space to choose a value from particles stored in memory or those closest to the best vibrating particle.

2.2. VPS flowchart

This section provide the flowchart of how the Vibrating Particles System (VPS) operate to solve different optimizations problems. It can be seen that the flowchart consists of three main steps:

- Initialization: in this step all the VPS attributes (maxNEFs, p , nVP, α ,

$vpmcr$, ω_1 , ω_2 , and ω_3) are initialized.

- Algorithm Body: in this step the optimization starts and the values of (D, HP, GP, BP, Fit, PFit, Fit_M, PFit_M, NFE, and maxNFE) are updated until the stop condition is satisfied.
- Results: after optimizing the result (PFit) this step define the way to display the optimal results.

Fig. 6-2, shows the VPS flowchart.

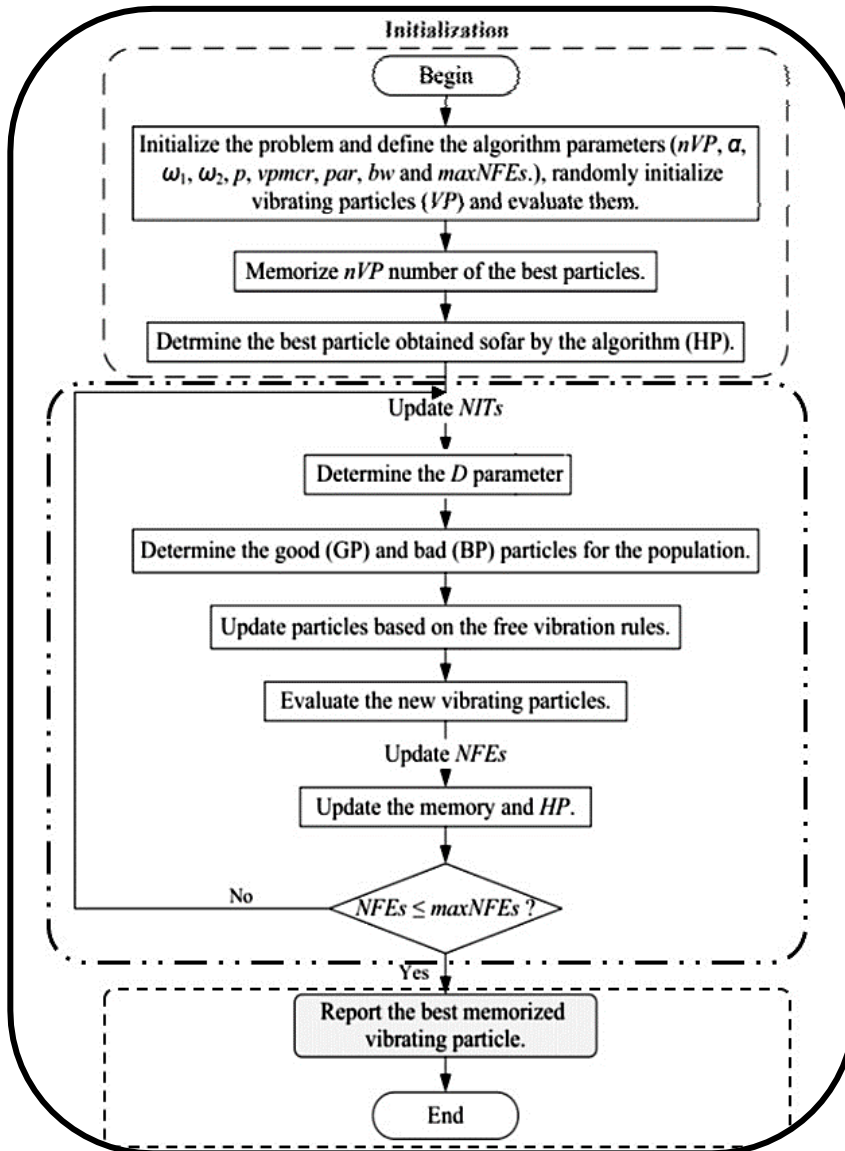


Fig. 6-2:VPS flowchart

3. Modifications of VPS

In general, all metaheuristics algorithms undergo several adjustments and enhancements after their first appearance so that they may be utilized to tackle diverse issues.

Following the release of the Vibrating Particles System (VPS) algorithm in 2017, various adjustments were made to the original VPS to increase the performance of the proposed algorithm.

Some of the adjustments and enhancements to the VPS algorithm are listed and organized in this part by development year (Almufti, 2022b), as show in Table 6-1.

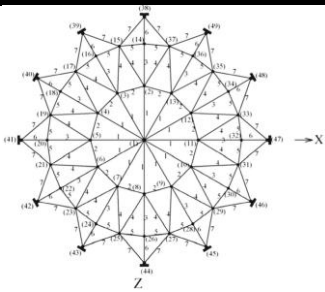
Table 6-1: VPS based algorithms

#	Abbr.	Name	Author	Year	Ref.
1.	VPS	Vibrating particles system	Kaveh et al.	2017	(Kaveh & Ghazaan, 2017)
2.	EVPS	Enhanced Vibrating Particles System	Patrick et al.	2019	(Gnetchejo, 2019)
3.	MO-VBPSO	Multi-Objective Vibration-Based Particle-Swarm-Optimized	Liang Ou et al.	2020	(Ou, Zeng, Chang, & Lin, 2020)

4. VPS Applications

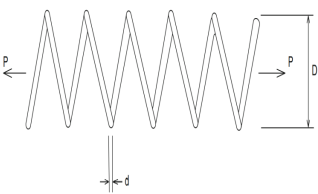
Over the years, the Vibrating Particles System (VPS) algorithm and its variations have demonstrated good performance in addressing several real-world issues, and it has been utilized to solve limited, single goal, and multi-objective problems in different areas (Almufti, 2022c), as shown in table 6-2.

Table 6-2: Summary of VPS applications in different field

#	Application	Discussion	Image	Ref.
1.	optimum design of truss structures	In this method, the solution candidates are modelled as particles that progressively approach their equilibrium locations. Equilibrium locations are obtained by combining current population and historically optimal places in order to create a good balance between exploration and exploitation.		(Kaveh & Ghazaan, 2017)

2. tension/compression spring design problem

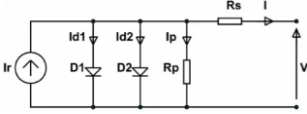
This approach is presented to converge a tension/compression spring design problem (TCSDD) solution to an optimum solution. The mechanical phenomena of vibration creates oscillations around an equilibrium condition. It's one of the most well-known Single-Objective Constrained Optimization Problems (SOCOP) in engineering.



(Kaveh & Bakhshpoor, 2019)

3. Parameters Estimation of Photovoltaic System

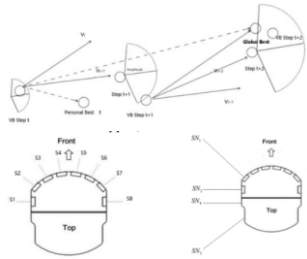
Several characteristics must be retrieved from the solar panel in order to evaluate its performance. These characteristics are critical for evaluating, monitoring, and optimizing solar systems. Enhanced vibrating particles system is one of the approaches created to extract photovoltaic characteristics from current-voltage (I-V) characteristic curves. It is inspired by free vibration of single degree of freedom systems with viscous damping.



(Gnetchejo, 2019)

4. Boundary-Following of Mobile-Robot Simulation Environment

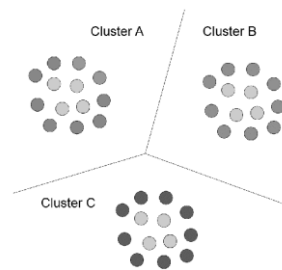
This chapter presents a multi-objective vibration-based particle-swarm-optimization (MO-VBPSO) algorithm with enhanced exploration ability and convergence performance, for training fuzzy-controller (FC) to achieve robot control.



(Ou, Zeng, Chang, & Lin, 2020)

5. Hard Clustering Problems

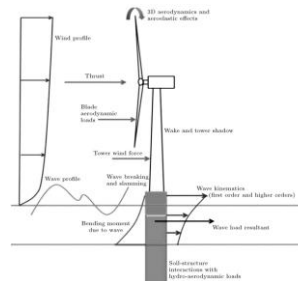
In the field of data analysis, clustering is an unsupervised technique that can be used to find identical sets of data. But, it is tough task to find the optimal centroid for a given dataset, especially in hard clustering problems. Recently, a vibrating particle system (VPS) algorithm was developed for solving the optimization problems.



(Parmar, Kumar, Singh, & Singh, 2019)

6. monopile offshore wind turbine structures

Considering both size and dimensions of the offshore wind turbine structures, design optimization of such structures is a fruitful yet, simultaneously, onerous task due to the tempestuous complexity of the problem, which mostly comes from their environment



(Kaveha & Sabeti, 2019)

5. Conclusion

Vibrating Particles System (VPS) algorithm is a Physics-based metaheuristic algorithm proposed in 2017 by Kaveh et al. (Kaveh & Bakhshpoor, Metaheuristics: Outlines, MATLAB Codes and Examples, 2019). After its appearance, many modifications has been proposed on it and it has been adapt to solve various problem in different fields. This chapter firstly addressed this overviewed the original VPS algorithm and then it presented some of its modifications were presented, finally some of its application considering were discussed. The applications in different fields including engineering optimization etc.

6. References

- A. Kaveh, V. M. (2015). Colliding Bodies Optimization. springer.
- Abualigah, L. (2019). Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering. Studies in Computational Intelligence. doi:10.1007/978-3-030-10674-4_2
- Almufti, S. M. (2015). U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem. Retrieved from Hdl.handle.net
- Almufti, S. M. (2019). Historical survey on metaheuristics algorithms. International Journal of Scientific World, 7(1), 1-12. doi:10.14419/ijsw.v7i1.29497
- Almufti, S. M., Marqas, R. B., Othman, P. S., & Sallow, A. B. (2021). Single-based and Population-based Metaheuristics for Solving NP-hard Problems. Iraqi J Sci, 62(5), 1-11.

- Celik, Y., & Kutucu, H. (2018). Solving the Tension/Compression Spring Design Problem by an Improved Firefly Algorithm. Retrieved from <http://ceur-ws.org/Vol-2255/chapter2.pdf>
- Celik, Y., & Kutucu, H. (2018). Solving the Tension/Compression Spring Design Problem by an Improved Firefly Algorithm.
- Deb, S., Fong, S., & Tian, Z. (2015). Elephant Search Algorithm for optimization problems. Tenth International Conference on Digital Information Management (ICDIM).
- GENC, H. M., EKSİN, I., & EROL, O. K. (2013). Big bang-big crunch optimization algorithm with local directional moves. *Turkish Journal of Electrical Engineering & Computer Sciences*, 21, 1359 – 1375. doi:10.3906/elk-1106-46
- Gnetchejo, P. J. (2019). Enhanced Vibrating Particles System Algorithm for Parameters Estimation of Photovoltaic System. *Journal of Power and Energy Engineering*, 7, 1-26.
- Ihsan, R. R., Almufti, S. M., Ormani, B. M., Asaad, R. R., & Marqas, R. B. (2021). A Survey on Cat Swarm Optimization Algorithm. *Asian Journal of Research in Computer Science*, 10(2), 22-32.
- Kaveh, A., & Bakhshpoor, T. (2019). *Metaheuristics: Outlines, MATLAB Codes and Examples*. Springer.
- Kaveh, A., & Farhoudi, N. (2013). A new optimization method: Dolphin echolocation. *Advances in Engineering Software*, 59, 53–70.
- Kaveh, A., & Ghazaan, M. I. (2017). A new meta-heuristic algorithm: Vibrating particles system. *Scientia Iranica*, 24(2), 551-566.
- Kaveh, A., & Ghazaan, M. I. (2017). Vibrating particles system algorithm for truss optimization with multiple natural frequency constraints. *Acta Mech*. doi:10.1007/s00707-016-1725-z
- Kaveha, A., & Sabeti, S. (2019). Optimal design of monopile offshore wind turbine structures using CBO, ECBO, and VPS algorithms. *Scientia Iranica*, 26(3), 1232-1248.
- Lobato, F. S., & Jr., V. S. (2014). Fish swarm optimization algorithm applied to engineering system design. *Latin American Journal of Solids and Structures*, 11(1). doi: <https://doi.org/10.1590/S1679-78252014000100009>
- Marqas, R. B., Almufti, S. M., Ahmed, H. B., & Asaad, R. R. (2021). Grey wolf optimizer: Overview, modifications and applications. *International Research Journal of Science, Technology, Education, and Management*, 1(1), 44-56.
- Ou, L., Zeng, G., Chang, Y.-C., & Lin, C.-T. (2020). Multi-Objective Vibration-Based Particle-Swarm-Optimized Fuzzy Controller With Application to Boundary-Following of Mobile-Robot Simulation Environment. 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC).
- Parmar, A., Kumar, Y., Singh, P. K., & Singh, V. (2019). Vibrating Particle System Algorithm for Hard Clustering problem. *SCIENCE & TECHNOLOGY*, 27(2), 815 - 827.
- Rao, R. V. (2016). *Teaching Learning Based Optimization Algorithm*. Springer.
- Sheta, A., Faris, H., Braik, M., & Mirjalili, S. (2020). Nature-inspired metaheuristics search algorithms for solving the economic load dispatch problem of power system: a comparison study. In *Applied nature-inspired computing: algorithms and case studies* (pp. 199-230). Springer, Singapore.
- Tabrizian, Z., Amiri, G. G., & Beigy, M. H. (2014). Charged System Search Algorithm Utilized for Structural Damage Detection. *Shock and Vibration*. doi:10.1155/2014/194753

- Uymaz, S. A., & Tezel, G. (2014). Cuckoo Search (CS) Optimization Algorithm for Solving Constrained Optimization Problems. International Conference on Computer Science, Engineering and Technology.
- Zebari, A. Y., Omer, H. K., & Almufti, S. M. (2019). A comparative study of particle swarm optimization and genetic algorithm. *Journal of Advanced Computer Science & Technology*, 8(2), 40-45. doi:10.14419/jacst.v8i2.29401
- Almufti, S. (2022a). Vibrating Particles System Algorithm: Overview, modifications and applications. *ICONTECH International Journal*, 6(3), 1–11. <https://doi.org/10.46291/icontechvol6iss3pp1-11>
- Almufti, S. M. (2022b). Vibrating Particles System Algorithm performance in solving constrained optimization problem. *Academic Journal of Nawroz University*, 11(3), 231–242. <https://doi.org/10.25007/ajnu.v11n3a1499>

Chapter 7: Social Spider optimization Algorithm

1. Introduction

With the fast growing in the complexity of modern optimization problems. Depending on the nature of phenomenon simulations is becoming increasingly attractive as an efficient tool for optimization, Swarm intelligence (SI) is a field that inspired by the collective behaviour of real insects or animals swarms in the natural (Almufti, 2017). In the past, many algorithms developed base on these behaviours such as (ACO, BSC, EHO, BA, HHO, HS, DE, EA, ABC,....etc.) solve a wide range of complex optimization problems. Bonabeau defined swarm intelligence as “any attempt to design algorithms or distributed problem solving devices inspired by the collective behaviour of the social insect colonies and other animal societies” (Bonabeau, et al., 1999). In this chapter, Social Spider Optimization (SSO) which is a swarm based metaheuristics algorithm is reviewed. The Social Spider Optimization algorithm is founded base on the simulation of cooperative behaviour in social-spiders colony (Jain, Singh, & Rani, 2019). In the proposed algorithm, a group of spiders which interact with each other by the biological laws of the cooperative colony (Almufti, 2021). SSO algorithm requires two different search agents (spiders): males and females. Based on gender, each individual is steered by a set of different evolutionary-operators which mimic different cooperative behaviours that are naturally seems in the colony (Jain, Singh, & Rani, 2019), (Cuevas et al., 2013). SSO has been widely modified and applied in several fields. This article presents a review of the Social Spider Optimization (SSO) and its modifications.

2. Social Spider optimization Algorithm:

The Social Spider Optimization (SSO) proposed by Erik Cuevas et al., in 2013 (Cuevas et al., 2013). It is a population-based Swarm intelligent algorithm that inspired by the natural cooperative behaviour of the social spider in the colony. Spider Colonies are formed mainly by two elements: spiders and communal web. Web is represented by the

searching field domain while problem solutions are represented by the insects. SSO considers two search agents (spiders): male and female. Each individual is directed by a different set of evolutionary rule depending on gender which mimics different cooperative behaviours typically found in the colony. This individual categorization allows reducing critical flaws present in several SI approaches such as incorrect exploration exploitation balance and premature convergence(Yu & Li, 2015).

In nature, heavier individuals dominate the lighter ones. This behaviour is copied to the algorithm and the spider's weight is proportional to the solution evaluation. Spiders are divided by gender and each one have a different behaviour in the colony. This difference is implemented using unique evolutionary operators for males and females. Gender balance on the colony is normally around 70% of females. The classical SSO algorithm starts by populating randomly the first generation with uniform distribution in the search space. A gender is addressed to each individual. The first agents (spiders) on the population matrix are addressed feminine and the rest as masculine (Singh, 2021).The cutting point is given by Equ.(1) :

$$N_f = \text{floor}[(0.9 - r * 0.25) N] \quad (1)$$

where N_f is the number of females, N is the population size, floor rounds each element to the nearest integer, and r is a random number in the unitary range $[0,1]$. All elements are then evaluated on the objective function and the best solution (spider) and best objective function are recorded. After the initialization process the algorithm starts the searching loop that only ends when the maximum number of function evaluations or the target function value is reached. The first step in the searching loop is to calculate the spiders weight. This calculation by Equ. (2):

$$w_i = 1 - \frac{FS_i - FS_{best}}{FS_{worest} - FS_{best}} \quad (2)$$

where w_i is the weight for the i_{th} spider, FS_i is the objective function value for the i_{th} spider, FS_{best} is the best objective value in the population and FS_{worest} is the worst objective value reached. The communal web is the natural substrate where all spiders live, exchanging information according to their distance and weight. There are mainly three types of communication among spiders in the web known as Vibci , Vibfi, Vibbi can be shown in Fig. 7-1, and can be calculated by Equ. (3) (Luque-Chang et al., 2018).

$$\text{Spider communication Types} \begin{cases} Vibci_i = w_c * \exp\left(\frac{-d_{i,c}}{d_{coeff}}\right) \\ Vibfi_i = w_{cf} * \exp\left(\frac{-d_{i,cf}}{d_{coeff}}\right) \\ Vibbi_i = w_b * \exp\left(\frac{-d_{i,b}}{d_{coeff}}\right) \end{cases} \quad (3)$$

Where:

I	Is the i_{th} individual spider
C	is the closest heavier member to the i_{th} element.
Cf	is the closest female to the i_{th} individual.
Vibci	it represents the vibration perceived by the i_{th} spider and emitted by the individual c
Vibfi	it represents the vibration perceived by the i_{th} individual, emitted by the member cf
Vibbi	it represents the vibration perceived by the i_{th} spider that is emitted by the best spider in the web.
w_c	is the closest heavier member weight
$d_{i,c}$	is the Euclidian distance between the i_{th} and c individuals
dcoeff	is a adaptive factor.
w_{cf}	is the closest female weight
$d_{i,cf}$	is the distance between the i_{th} spider and its closest female
w_b	is the best solution's weight
$d_{i,b}$	is the distance between the i_{th} and the best individual

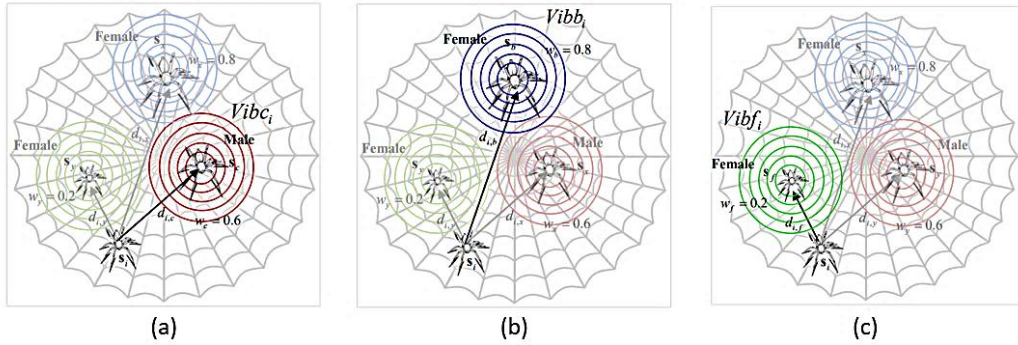


Fig. 7-1: Configuration of each special relation: a) Vibci , b) Vibbi and c) Vibfi

spiders are moved either on a repulsion movement or on an attraction. This decision is made by attributing a random unitary number, to each female individual with respect to pf, which is a constructive (threshold) parameter (Cuevas & Cienfuegos, 2014), it can be calculated by Equ. (4).

$$f_i = \begin{cases} f_i - \alpha \cdot \theta \cdot vibci \cdot (sc - f_i) - \beta \cdot \theta \cdot vibbi \cdot (sb - f_i) - \theta \cdot \left(\gamma - \frac{1}{2} \right) & \text{if } f_i > fp \\ f_i + \alpha \cdot \theta \cdot vibci \cdot (sc - f_i) + \beta \cdot \theta \cdot vibbi \cdot (sb - f_i) + \theta \cdot \left(\gamma - \frac{1}{2} \right) & \text{if } f_i < fp \end{cases} \quad (4)$$

where f_i is the i_{th} female, sc represents the closest heavier spider coordinates, sb is the best spider's position, α , β , γ are unitary-ranged random numbers generated with uniform distribution and θ is an adaptive factor.

Male spiders are moved according to dominance. The dominance is given by the males median weight's. Males lighter than the median are consider non-dominant and moved according to Equ. (5):

$$m_i = m_i + \alpha \cdot \theta \left(\frac{\sum_{h=1}^{Nm} m_h \cdot w_{Nf+h}}{\sum_{h=1}^{Nm} w_{Nf+h}} - m_i \right) \quad (5)$$

where m_i represents the i_{th} male position, m_h is the h_{th} male position, Nm is the total number of males, w_{Nf+h} is the weight for the h_{th} male and α is an unitary-ranged random number[23]. Dominant males are those heavier than the median and are moved according to Equ. (6):

$$m_i = m_i - \alpha \cdot \theta \cdot vibfi \cdot (sf - m_i) + \theta \cdot \left(\gamma - \frac{1}{2} \right) \quad (6)$$

where m_i is the i_{th} male position, sf is the closest female to i_{th} male and α, γ are unitary-ranged random numbers.

Finally, after move all males and females on the web, the last operator is representing the mating behaviour where only dominant males will participate. The code will check if there is any female closer than radius of mating to a dominant male. The radius of mating is given by Equ.(7):

$$rm = \frac{\sum_{d=1}^D (p_d^h - p_d^l)}{2D} \quad (7)$$

where rm is the mating radius, p_d^h and p_d^l are respectively the upper and lower bound for a given dimension and D is the problem dimension. Males and females which are under the mating radius generate new candidate spiders according to the roulette method. Each candidate spider is evaluated in the objective function and the result is tested against all the actual population members(Jain, Singh, & Rani, 2019). If any member is worse than a new candidate, the new candidate will take the actual individual position assuming actual individual's gender. The classical SSO algorithm can be summarizes by the following flowchart Fig.7-2.

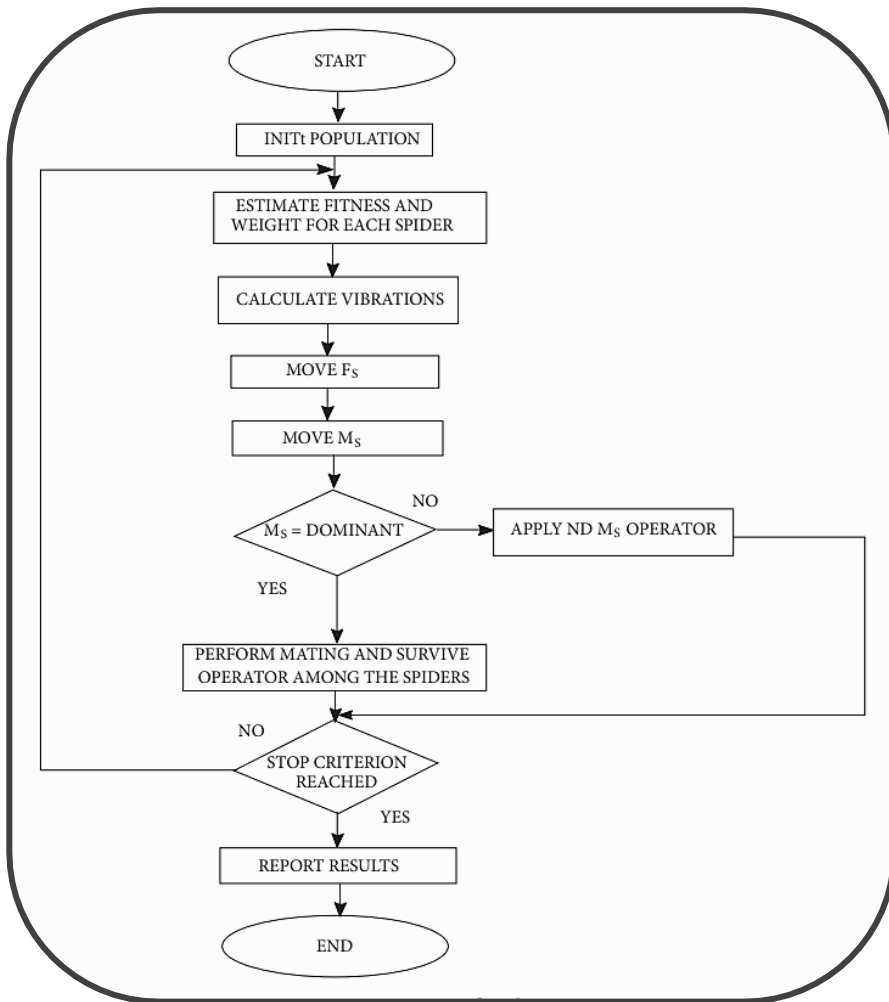


Fig. 7-2: SSO flowchart.

3. Social Spider optimization Algorithm Modifications:

After the appearance of Social Spider Optimization (SSO) algorithm in 2013 (Cuevas et al., 2013), many modifications has been applied to the classical SSO to improve the performances of the proposed algorithm.

In this section, the modifications and improvements of the SSO algorithm are reviewed.

3.1. Social Spider Optimization Based on Rough Sets (SSORS):

At the end of 2016, Mohamed Abd El Aziz et al. proposed a modification in Social Spider Optimization by adding the rough-set model for evaluating the fitness function to improve the performance of SSO, the proposed method called Social Spider Optimization Based on Rough Sets (SSORS) and has been adapt to solve

the minimum attribute reduction problem.

In the algorithm, the fitness function depends on the rough sets dependency degree and takes into consideration the number of selected features. The algorithm starts by randomly initializing a population of spiders. Next, each individual is converted into a binary vector of length N by using the following Equ.(8 and 9) [25]:

$$FP = \frac{1}{1 + e^{-x_i^j(t)}} \quad (8)$$

$$x_i^j(t+1) = \begin{cases} 0 & , \quad \text{if } x_i^j(t) > \epsilon \\ 1 & , \quad \text{other case} \end{cases} \quad (9)$$

where $x_i^j(t)$ is the spider value at the iteration t and $\epsilon \in [0, 1]$. The algorithm uses the dependency degree given by Equ.(10):

$$\gamma_c(D) = \frac{|POS_c(D)|}{|U|} \quad , \quad A = C \cup D \quad (10)$$

Where C and D are called condition and decision features and POSC(D) is the positive region that contains all the objects of U that can be classified in classes of U/D using information of C, which is used to evaluate the performance of the solutions. Furthermore, the fitness function is defined by Equ.(11) [25].

$$F(R) = p\gamma_R(D) + (1 - p) \left(1 - \frac{|R|}{|C|} \right) \quad (11)$$

where $P \in [0, 1]$ is a parameter that gives balance between number of selected features and the classification quality and $|\cdot|$ is the length of the feature set. The fitness of each spider is compared with the global best (Fbest), and if it has a better fitness value, then Fbest is replaced with the current spider and its position becomes the reduct set R. The process is repeated until a stop criterion is satisfied. Fig. 7-3 shows the SSORS computational procedure in form of a flowchart.

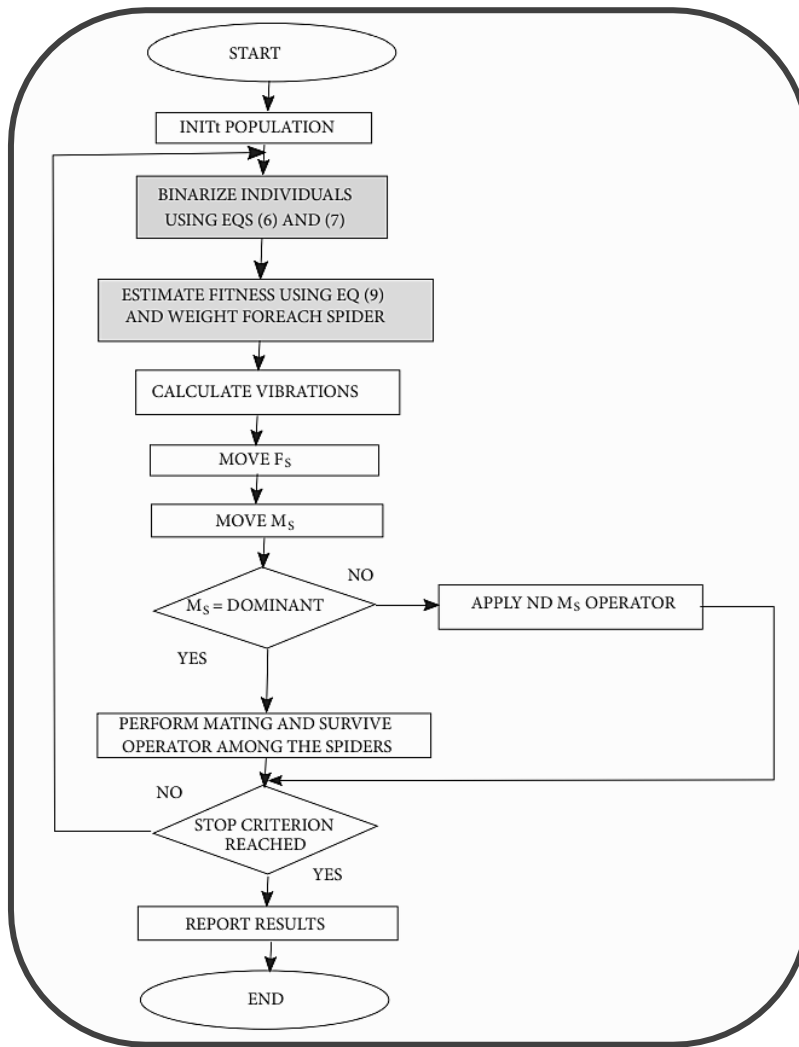


Fig. 7-3: SSORS flowchart

3.2. Simplex Method Social Spider Optimization (SMSSO):

An obstacle for the SSO to be applied in complex problems is its high computational cost. Therefore, as the dimensionality of a search space and the data amount increases, a problem of local optima entrapment and poor convergence rates is present. To overcome these problems, Yongquan Zhou et al. proposed to apply the simplex method to the original SSO (SMSSO) in 2017, enhancing the algorithm global and local search abilities and avoiding local optima entrapment while increasing the convergence rate.

The simplex method was introduced by Spendley et al (Spendley, Hext, & Himsworth, 1962), and is defined by some points equal to the number of dimensions in the search space plus one. The process of the simplex method is

described as follows:

Evaluate the solutions of the entire population and select the global best s_{best1} and the second best s_{best2} , assuming that s_{best1} is the spider that must be replaced and that $f(s_{best1})$, $f(s_{best2})$, and $f(s_r)$ are the corresponding fitness values.

Calculate the middle position (s_m) of s_{best1} and s_{best2} defined by:

$$s_m = \left(\frac{s_{best1} + s_{best2}}{2} \right) \quad (12)$$

Determine the refraction point s_{ref} given by

$$s_{ref} = s_m + \alpha(s_m - s_r) \quad (13)$$

The refraction coefficient α is typically set to one.

Compare the fitness value between s_{ref} and s_{best1} . If $f(s_{ref}) < f(s_{best1})$, the extension operation was performed based on the next equation

$$s_e = s_m + \gamma(s_{ref} - s_m) \quad (14)$$

where γ is the extension coefficient, and this parameter is usually set in two.Ten compare the fitness value of the extension point and the global best. If $s_e < s_{best1}$, s_r should be replaced by s_e and, in another case, s_{ref} will be substituted for s_r .

Compare the fitness values of s_{ref} and s_r . If $f(s_{ref}) < f(s_r)$, the compression operation should be performed using the following equation:

$$s_{comp} = s_m - \beta(s_r - s_m) \quad (15)$$

β is the condense coefficient; it is typically set to 0.5. Then, it is compared to the fitness values between s_{comp} and s_r , and if $f(s_{comp}) < f(s_r)$, s_r should be replaced with s_{comp} ; otherwise, s_{ref} will be substituted with s_r .

If $f(s_{best1}) < f(s_{ref}) < f(s_r)$, the condense point s_{cond} must identify performing shrink operations where the shrink coefficient is described by σ :

$$s_{cond} = s_m - \sigma(s_r - s_m) \quad (16)$$

If $f(s_{cond}) < f(s_r)$, s_r will be replaced with s_{cond} . In other case, s_{ref} will be replaced with s_r .

The proposed SMSSO algorithm is shown in Fig. 7-4 as a flowchart (Ali, 2007)

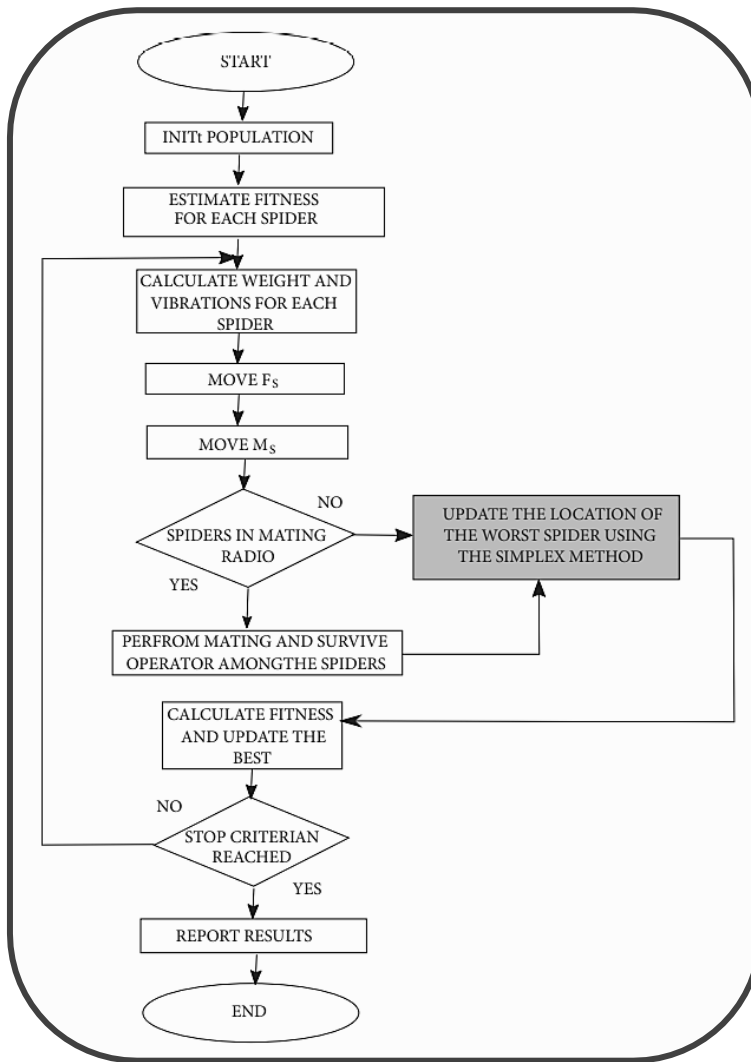


Fig. 7-4: SMSSO flowchart

3.3. Modified SSO Approach Based on Beta Distribution and Natural Gradient Local Search (MSSO):

Metaheuristic algorithms need a right balance between exploration and exploitation to give successful results in optimization problems. The classical SSO requires the random selection of parameters α , β , and θ in Equ.(4) and (5)) to control the movement of the spiders, which can affect the mentioned balance leading the algorithm to a premature convergence. With the aim of improving this balance, Carlos E. Klein et al. proposed a modification for the SSO (MSSO) in 2016 (Shukla & Nanda, 2016), where the mentioned parameters are selected from a Beta distribution in the range [0-1] instead of the use of random numbers. The use of this distribution helps to preserve diversity and avoid premature

convergence, improving the algorithm exploration. Furthermore, to improve the exploitation the author proposed the use of natural gradient (NG) with a rank one covariance matrix approximation to local search in each generation. In this method, the best spider realizes local search using NG after performing the operator (Mahato & Singh, 2018). The NG parameters are the learning rate for the mean value and for the scale factor. The complete computational process is summarized in Fig. 7-5 as a flowchart. To prove the performance of the MSSO algorithm, authors considered the solution of two engineering problems such as the solenoid and brushless motor design.

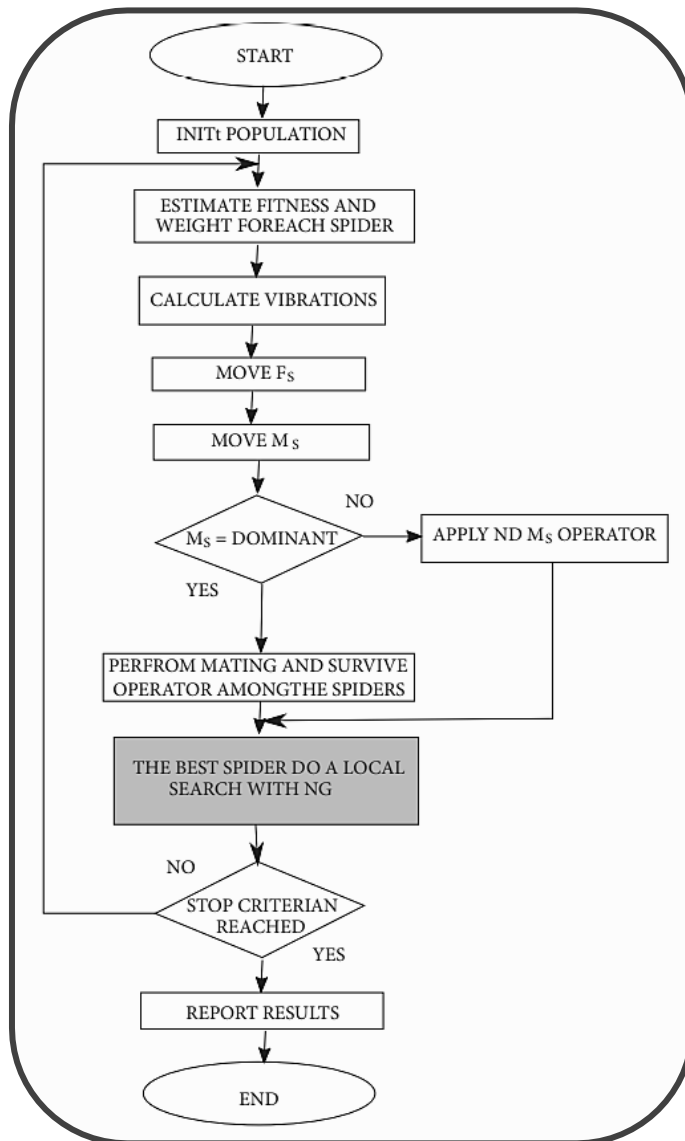


Fig. 7-5: MSSO flowchart.

3.4. Elite Opposition-Based Social Spider Optimization (EOSSO):

The SSO emulates the cooperative behaviour of a spider colony by stochastic position changes; with this method the probability of getting a good solution is relatively low. With the purpose to increase the probability of getting a better solution Ruxin Zhao et al. proposes the use of Elite Opposition-Based Learning Strategy EOLS in the SSO and creates the EOSSO in 2017 (Abd El Aziz & Hassanien, 2017). Opposition-Based Learning (OBL) is a machine intelligence strategy that considers the current individual and its opposite particle at the same time to get a better approximation for the current candidate solution. It has been proved that an opposite candidate has a greater chance to be closer to the global optimal than a random candidate solution. Some of the concepts of OBL are defined as follows. The EOSSO main idea is to calculate and evaluate the opposite solution of each particle at the same time and select the better one as an individual for the next generation. The best fitness valued individual is seen as an elite individual. At the global optimization process, the strategy expands the search space of the algorithm and strengthens the diversity of the population. Thus the global searching ability can be enhanced and help to get a better optimal solution (Abd El Aziz & Hassanien, 2017). The complete computational process is summarized in Fig. 7-6.

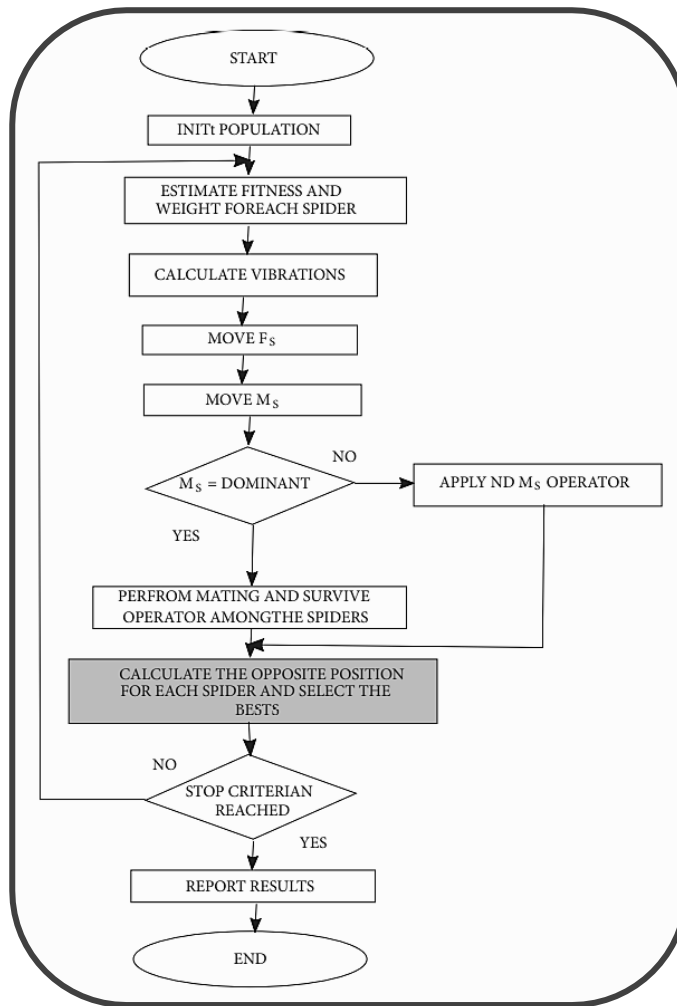


Fig. 7-6:EOSSO flowchart

3.5. Parallel Social Spider Clustering Algorithm for High Dimensional Data sets (P-SSO):

As the dimensionality in specific problems increases, it is desirable that the execution time of SI algorithms would be reduced. With this aim, Urvashi Prakash Shukla et al. suggested a parallel version of the SSO (P-SSO) in 2016 . In this version, the position of each spider (female, dominant and nondominant male) is updated simultaneously; this increases the computational speed, giving the algorithm the ability to work in high dimensional problems. The computational procedure of P-SSO is illustrated in Figure 9 as a flowchart (Zhou, Zhou, Luo, & Abdel-Basset, 2017). Tis algorithm was applied to solve clustering problems with high dimensional data. According to the authors, the approach performs ten times faster than the original SSO algorithm. The P-SSO outperforms several clustering schemes even in other real-life applications as multispectral image segmentation

as a clustering problem. In terms of accuracy, the P-SSO provides two times better precision than one of its competitors fig.7-7 (Zhou, Zhou, Luo, & Abdel-Basset, 2017).

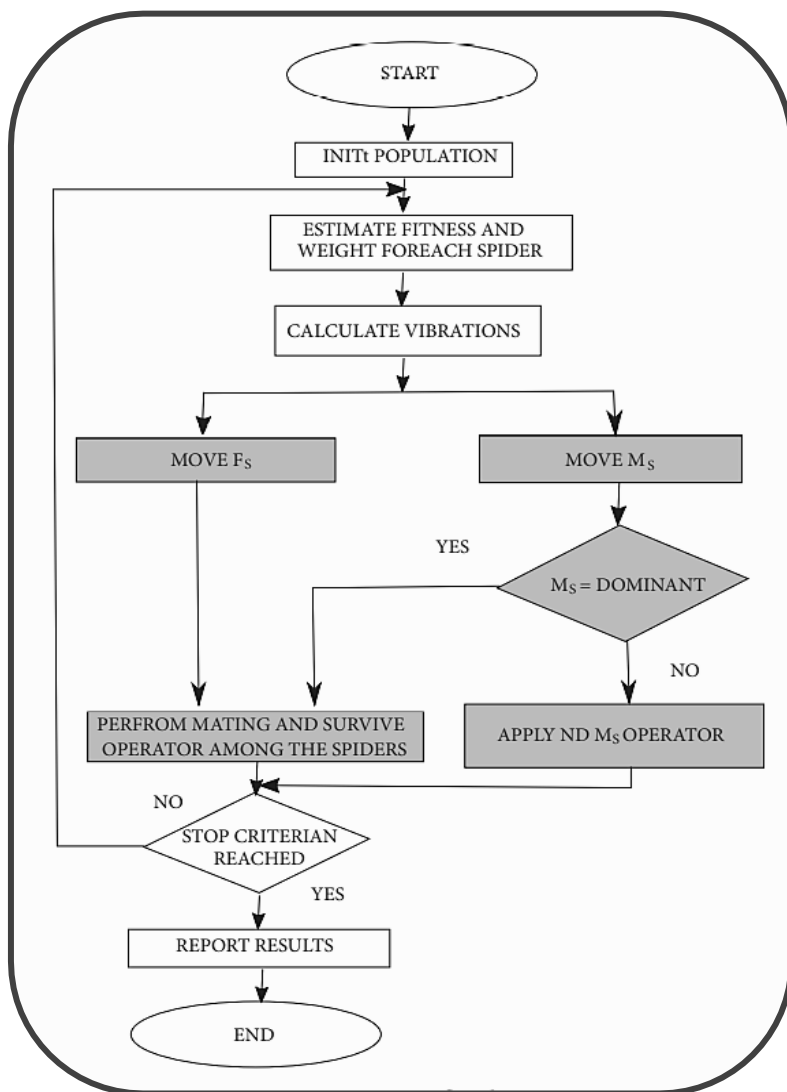


Fig. 7-7: P-SSO flowchart

3.6. Hybrid Social Spider Optimization and Genetic Algorithm (HSSOGA):

A desirable property for any population-based optimization technique is to avoid premature convergence and entrapment into local optima; with this aim, Tawhid and Ali proposed a hybrid optimization approach based on both SSO and GA algorithms in 2016 (Zhou, Zhao, Luo, & Wen, 2017). In that approach, the authors combine the properties of exploration and exploitation of the SSO. Ten, it applies the arithmetical crossover and mutation operators of the GA, calling the algorithm

Hybrid Social Spider Optimization and Genetic Algorithm (HSSOGA). With this combination, the search process is accelerated and helps to find a near optimal solution in a reasonable time (Almufti & Shaban, 2018), (Zhou, Zhao, Luo, & Wen, 2017). The HSSOGA algorithm consists of mainly three steps:

- Apply the social spider cooperative operators to take advantage of its balance between exploration and exploitation.
- Subdivide the population and apply the arithmetical crossover operation on the populations with the aim of improving the search diversity of the proposed algorithm.
- Apply the GA mutation operator in order to avoid premature convergence.

Fig. 7-8 presents the computational procedure of HSSOGA as a flowchart.

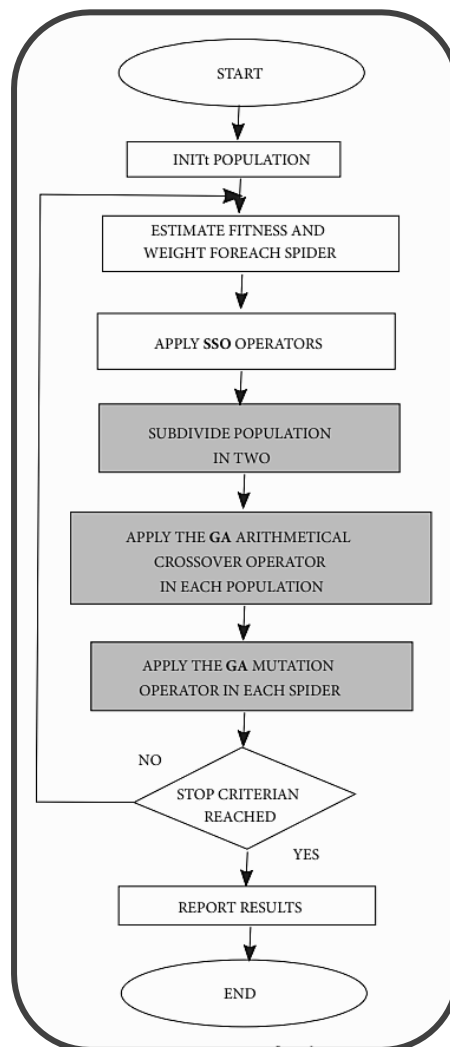


Fig. 7-8: HSSOGA flowchart

3.7. Improved Social Spider Optimization (ISSO):

With the objective of accelerating the convergence efficiency and search ability of the SSO, Shuang-Cheng Sun et al. (Khorramnia et al., 2015) proposed five improved SSO algorithms in 2017. The first algorithm ISSO1 takes advantage of the historical process adding the historical best position to operate the movement of the spiders in the search area through a vibration perceived by the i -th spider, leading to an increment of the convergence speed. The second improvement ISSO2 tries to overcome the chaotic search on the earlier stages of the search process due to the low influence of the spider's vibrations on a high distance between them, adding acceleration coefficients to the half of the female and dominant spiders. Furthermore, a random search term that decreases with each iteration and is controlled by an inertia weight is added. A small step size causes a careful search among the spiders but decreases the convergence speed. On the other hand, a big step size increases the convergence speed but decreases the search accuracy, which may lead to losing the optimal position (Khorramnia et al., 2015). To solve these troubles in the ISSO3 algorithm add a linearly decreased step size leading this to a reasonable tradeoff between computational time and search accuracy. In the ISSO4 algorithm, a mutation operator for the nondominant spiders is added with the objective to improve the solutions diversity. Finally, the ISSO5 algorithm makes use of the improvements of the other four algorithms at the same time. Fig. 7-9 presents the computational process of each version ISSO1, ISSO2, ISSO3, and ISSO4 in form of a flowchart (Almufti & Shaban, 2018).

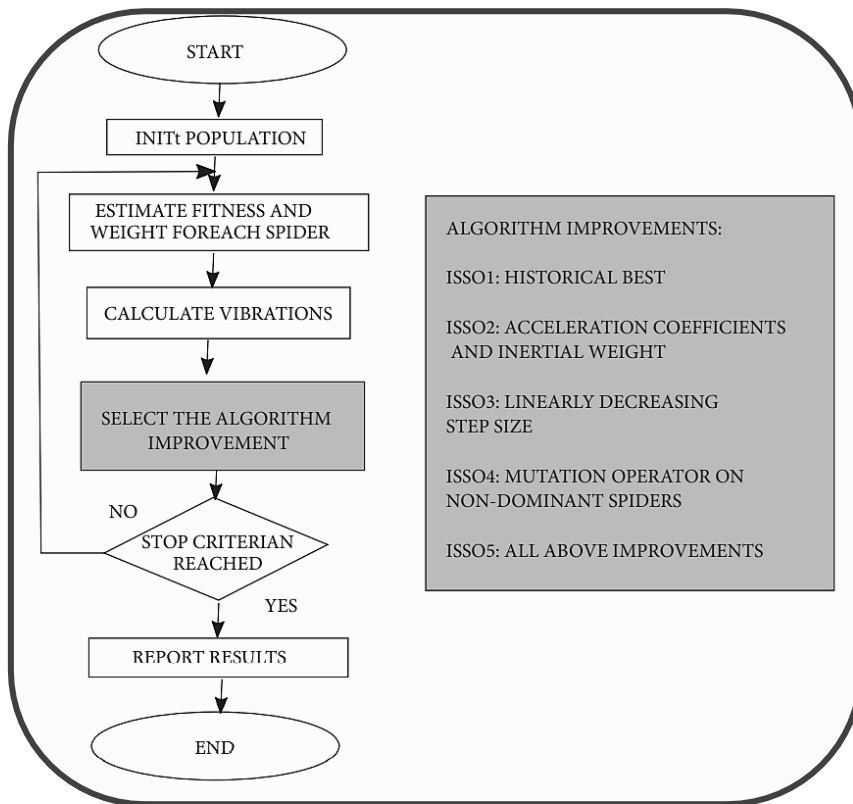


Fig. 7-9: ISSO flowchart

4. Applications:

Since its introduction, SSO and its modified algorithms has potential applied in a wide to solve various problems in several fields. This section presents a review of some of the most important problems.

Table 7-1: Applications of SSO

#	Application Domain	Reference
1.	Binary optimization problems (BOPs)	(Cuevas, Osuna, & Oliva, 2017)
2.	Wireless sensor networks (WSN)	(Fausto et al., 2019)
3.	Multiple-Pursuer Multiple-Evader (MPME)	(Husodo et al., 2020)
4.	Performance of Multiple Pursuer Drones in Neutralizing Attacks	(Husodo et al., 2020)
5.	Optimal reactive power dispatch problem	(Nguyen & Vo, 2019)
6.	Clustering for high-dimensional data sets	(Shukla & Nanda, 2016)
7.	Feedforward Neural Networks (FNN)	(Tawhid & Ali, 2017)
8.	Vector Machines Parameters Tuning	(Tawhid & Ali, 2017)
9.	Parameter improvement in the Lukasiewicz structure	(Tawhid & Ali, 2017)

10. Image fusion for contrast and brightness preservation (Tawhid & Ali, 2017)
 11. Image enhancements (Ouadfel & Taleb-Ahmed, 2016)
 12. Feature selection for optimal Lukasiewicz parameters (Husodo et al., 2020)
 13. Image segmentation via multilevel image thresholding (El-Fergany & El-Hameed, 2017)
 14. Template matching (TM) in image processing (Hejrati, Fattahi, & Faraji, 2014)
 15. Unscented Transform for Wind Turbine Uncertainty (Khorramnia et al., 2015)
 16. Sensor deployment scheme (Zhou, Zhou, Luo, & Abdel-Basset, 2017)
 17. Optimal congestion management (Zhou, Zhao, Luo, & Wen, 2017)
 18. Efficient frequency controllers for autonomous two-area system (Zhao, Luo, & Zhou, 2017)
 19. Tuning gains of a PID controller (Zhao, Luo, & Zhou, 2017)
 20. Optimal design of fractional controller for LFC (Yang, 2011)
-

5. Discussion:

Social Spider Optimization (SSO) is a population-based algorithm that inspired by the cooperative behaviour of the social spider proposed by Erik Cuevas et al. in 2013,. SSO considers two search agents male and female which are steered by a different set of evolutionary operators depending on its gender to verify a cooperative behaviours typically found in the spider colony. After its introduction, the SSO has been used to solve a wide variety of problem in computer and engineering fields.

6. References

- Almufti, S. (2017). Using swarm intelligence for solving NP-hard problems. *Academic Journal of Nawroz University*, 6(3), 46–50. <https://doi.org/10.25007/ajnu.v6n3a78>
- Almufti, S. M. (2019). Historical survey on metaheuristics algorithms. *International Journal of Scientific World*, 7(1), 1. <https://doi.org/10.14419/ijsw.v7i1.29>(Zhao, Luo, & Zhou, 2017)7
- Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. Oxford University Press.
- Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation*, 44, 148–175. <https://doi.org/10.1016/j.swevo.2018.02.013>
- Cuevas, E., Cienfuegos, M., Zaldívar, D., & Pérez-Cisneros, M. (2013). A swarm optimization algorithm inspired in the behaviour of the social-spider. *Expert Systems with Applications*, 40(16), 6374–6384. <https://doi.org/10.1016/j.eswa.2013.05.041>
- Cuevas, E., & Cienfuegos, M. (2014). A new algorithm inspired in the behaviour of the social-spider for constrained optimization. *Expert Systems with Applications*, 41(2), 412–425. <https://doi.org/10.1016/j.eswa.2013.07.067>
- Luque-Chang, A., Cuevas, E., Fausto, F., Zaldívar, D., & Pérez, M. (2018). Social spider optimization algorithm: Modifications, applications, and perspectives. *Mathematical Problems in Engineering*, 2018, 1–29. <https://doi.org/10.1155/2018/6843923>

- Singh, D. (2021). A new bio-inspired social spider algorithm. *International Journal of Applied Metaheuristic Computing*, 12(1), 79–93. <https://doi.org/10.4018/ijamc.2021010105>
- Yu, J., & Li, V. (2015). A social spider algorithm for global optimization. *Applied Soft Computing*, 30, 614–627. <https://doi.org/10.1016/j.asoc.2015.02.014>
- Zhao, R., Luo, Q., & Zhou, Y. (2017). Elite opposition-based social spider optimization algorithm for global function optimization. *Algorithms*, 10(1), Article 9. <https://doi.org/10.3390/a10010009>
- Tawhid, M. A., & Ali, A. F. (2017). A hybrid social spider optimization and genetic algorithm for minimizing molecular potential energy function. *Soft Computing*, 21(21), 6(Zhao, Luo, & Zhou, 2017)9–6514.
- Sun, S.-C., Qi, H., Ren, Y.-T., Yu, X.-Y., & Ruan, L.-M. (2017). Improved social spider optimization algorithms for solving inverse radiation and coupled radiation–conduction heat transfer problems. *International Communications in Heat and Mass Transfer*, 87, 132–146.
- Baş, E., & Ülker, E. (2020). A binary social spider algorithm for continuous optimization task. *Soft Computing*, 24(17), 12953–12979. <https://doi.org/10.1007/s00500-020-04718-w>
- Fausto, F., Cuevas, E., Maciel-Castillo, O., & Morales-Castañeda, B. (2019). A real-coded optimal sensor deployment scheme for wireless sensor networks based on the social spider optimization algorithm. *International Journal of Computational Intelligence Systems*, 12(2), 676. <https://doi.org/10.2991/ijcis.d.190614.001>
- Wang, G., Dos Santos Coelho, L., Gao, X., & Deb, S. (2016). A new metaheuristic optimisation algorithm motivated by elephant herding behaviour. *International Journal of Bio-Inspired Computation*, 8(6), 394. <https://doi.org/10.1504/IJBIC.2016.10002274>
- Almufti, Asaad, & Salim, B. (2019). Review on elephant herding optimization algorithm performance in solving optimization problems. *Sciencepubco.com*. <https://www.sciencepubco.com/index.php/ijet/article/view/28473>
- Zebari, A. Y., Almufti, S., & Abdulrahman, C. (2020). Bat algorithm (BA): Review, applications and modifications. *International Journal of Scientific World*, 8(1), 1. <https://doi.org/10.14419/ijsw.v8i1.30120>
- Assad, R. R., & Abdulnabi, N. (2018). Using local searches algorithms with ant colony optimization for the solution of TSP problems. *Academic Journal of Nawroz University*, 7(3), 1–6. <https://doi.org/10.25007/ajnu.v7n3a193>
- Almufti, S., & Shaban, A. (2018). U-turning ant colony algorithm for solving symmetric traveling salesman problem. *Academic Journal of Nawroz University*, 7(4), 45–(Zhao, Luo, & Zhou, 2017). <https://doi.org/10.25007/ajnu.v6n4a270>
- Spendley, W., Hext, G. R., & Himsforth, F. R. (1962). Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4), 441–461.
- Ali, M. M. (2007). Synthesis of the β -distribution as an aid to stochastic global optimization. *Computational Statistics & Data Analysis*, 52(1), 133–1(Zhao, Luo, & Zhou, 2017).
- Shukla, U. P., & Nanda, S. J. (2016). Parallel social spider clustering algorithm for high dimensional datasets. *Engineering Applications of Artificial Intelligence*, 56, 75–90.
- Mahato, D., & Singh, R. (2018). On maximizing reliability of grid transaction processing system considering balanced task allocation using social spider optimization. *Swarm and Evolutionary Computation*, 38, 202–217. <https://doi.org/10.1016/j.swevo.2017.07.011>

- Abd El Aziz, M., & Hassanien, A. (2017). An improved social spider optimization algorithm based on rough sets for solving minimum number attribute reduction problem. *Neural Computing and Applications*, 30(8), 2441–2452. <https://doi.org/10.1007/s00521-016-2804-8>
- Zhou, Y., Zhou, Y., Luo, Q., & Abdel-Basset, M. (2017). A simplex method-based social spider optimization algorithm for clustering analysis. *Engineering Applications of Artificial Intelligence*, 64, 67–82.
- Zhou, Y., Zhao, R., Luo, Q., & Wen, C. (2017). Sensor deployment scheme based on social spider optimization algorithm for wireless sensor networks. *Neural Processing Letters*, 1–24.
- Khorramnia, R., Akbarizadeh, M.-R., Jahromi, M. K., Khorrami, S. K., & Kavusifard, F. (2015). A new unscented transform for considering wind turbine uncertainty in ED problem based on SSO algorithm. *Journal of Intelligent & Fuzzy Systems*, 29(4), 1479–1.
- Cuevas, E., Osuna, V., & Oliva, D. (2017). *Evolutionary computation techniques: A comparative perspective* (Vol. 686). Springer.
- Nguyen, T., & Vo, D. (2019). Improved social spider optimization algorithm for optimal reactive power dispatch problem with different objectives. *Neural Computing and Applications*, 32(10), 5919–5950. <https://doi.org/10.1007/s00521-019-04073-4>
- Maurya, L., Mahapatra, P. K., & Kumar, A. (2017). A social spider optimized image fusion approach for contrast enhancement and brightness preservation. *Applied Soft Computing*, 52, 575–592.
- Dollaor, J., Chiewchanwattana, S., Sunat, K., & Muangkote, N. (2016). The application of social-spider optimization for parameter improvement in the Lukasiewicz structure. In *Proceedings of the 8th International Conference on Knowledge and Smart Technology (KST 2016)* (pp. 27–32).
- Ouadfel, S., & Taleb-Ahmed, A. (2016). Social spiders optimization and flower pollination algorithm for multilevel image thresholding: A performance study. *Expert Systems with Applications*, 55, 566–584.
- Husodo, A., Jati, G., Octavian, A., & Jatmiko, W. (2020). Enhanced social spider optimization algorithm for increasing performance of multiple pursuer drones in neutralizing attacks from multiple evader drones. *IEEE Access*, 8, 22145–22161. <https://doi.org/10.1109/ACCESS.2020.2969021>
- El-Fergany, A. A., & El-Hameed, M. A. (2017). Efficient frequency controllers for autonomous two-area hybrid microgrid system using social-spider optimiser. *IET Generation, Transmission & Distribution*, 11(3), 637–648.
- Hejrati, Z., Fattahi, S., & Faraji, I. (2014). Optimal congestion management using the social spider optimization algorithm. In *29th International Power System Conference*. Iran.
- Shayeghi, H., Molaei, A., & Ghasemi, A. (2016). Optimal design of FOPID controller for LFC in an interconnected multi-source power system. *International Journal on Technical and Physical Problems of Engineering*, 8(3), 36–44.
- Zhou, Y., Zhou, Y., Luo, Q., & Abdel-Basset, M. (2017). A simplex method-based social spider optimization algorithm for clustering analysis. *Engineering Applications of Artificial Intelligence*, 64, 67–82.
- Zhou, Y., Zhao, R., Luo, Q., & Wen, C. (2017). Sensor deployment scheme based on social spider optimization algorithm for wireless sensor networks. *Neural Processing Letters*, 1–24.

- Zhao, R., Luo, Q., & Zhou, Y. (2017). Elite opposition-based social spider optimization algorithm for global function optimization. *Algorithms*, 10(1), Article 9. <https://doi.org/10.3390/a10010009>
- Yang, X. (2011). Metaheuristic optimization. *Scholarpedia*, 6(8), 11472. [https://doi.org/10.42\(Zhao, Luo, & Zhou, 2017\)/scholarpedia.11472](https://doi.org/10.42(Zhao, Luo, & Zhou, 2017)/scholarpedia.11472)
- Almufti, S. (2021). The novel social spider optimization algorithm: Overview, modifications, and applications. *Icontech International Journal*, 5(2), 32–51.

Chapter 8: Cat Swarm Optimization Algorithm

1. Introduction

Metaheuristic algorithms have become indispensable tools for addressing complex, nonlinear, and high-dimensional optimization problems in science and engineering. Among these, Cat Swarm Optimization (CSO) has emerged as a notable swarm intelligence algorithm inspired by the natural behaviours of domestic cats, particularly their contrasting modes of rest (seeking mode) and movement (tracing mode). First introduced by Chu, Tsai, and Pan in 2006, CSO simulates the way cats alternate between observing their surroundings and actively chasing targets—behavioural dynamics that translate effectively into a robust search mechanism for both exploration and exploitation in computational search spaces (Ihsan, 2021).

This chapter provides a comprehensive introduction to the Cat Swarm Optimization algorithm, detailing its biological inspiration, mathematical formulation, and dual-mode search strategy that differentiates it from other nature-inspired techniques. The inherent simplicity, flexibility, and adaptability of CSO have led to its application across a wide array of domains, including feature selection, image processing, machine learning, wireless sensor networks, power system optimization, and structural engineering design (Almufti et al., 2019). Its ability to balance local search (via the seeking mode) with global exploration (via the tracing mode) has proven effective in handling multimodal and constrained optimization problems.

Over time, the original CSO algorithm has undergone significant enhancements to improve convergence speed, solution quality, stability, and applicability to dynamic and high-dimensional environments. Researchers have proposed numerous modifications, including adaptive parameter tuning, hybridization with other metaheuristics, multi-objective extensions, and problem-specific variants tailored to discrete, binary, and combinatorial optimization problems.

2. Cat Swarm Optimization

Cat Swarm Optimization is swarm intelligence algorithms introduced by Chu, Tsai, and Pan in the year 2006 for solving optimizations problems (Almufti et al., 2019) based on the behaviour of cats.

The original CSO algorithm was designed for solving a continuous and single-objective problems. In the nature, Cats are lazy animal and spend most of the time on rest. but, during the resting, Cats awareness is very high and they are conscious of what happens in around, and are continuously intelligently observing around them, and whenever they find target they start quickly moving towards that target. CSO algorithm has been designed based on combining these two characteristics of cats (Ihsan, 2021) (Agarwal & Mehta, 2014).

Cat Swarm Optimization algorithm consists of two modes: seeking and tracing modes (Ihsan, 2021). Each cat represents an agent has three main variables:

- **Position:** is a M-dimensions in the search space, and each dimension has its own velocity.
- **Fitness:** is a value shows how well the solution set (cat) is
- **flag:** is to classify the cats into either seeking or tracing mode

CSO should first specify how many cats should be engaged in the iteration and run them through the algorithm. To combine the two modes into the algorithm, a mixture ratio (MR) is defined. This parameter is chosen from the interval of [0, 1] and it determines what percentage of cats are in seeking mode and what percentage are in tracing mode. The best cat in each iteration is saved into memory until the stop criteria is achieved, and the one at the final iteration will represent the final solution(Sharafi et al., 2013).

2.1. Seeking Mode:

For modelling the behaviour of cats in resting time and being-alert, CSO corresponds the seeking mode. This mode is a time for thinking and deciding about next move. This mode has four main parameters which are mentioned as follow: seeking memory pool (SMP), seeking range of the selected dimension (SRD), counts of dimension to change (CDC) and self-position consideration (SPC) (Chu & Tsai, 2007), as shown in Fig. 8-1.

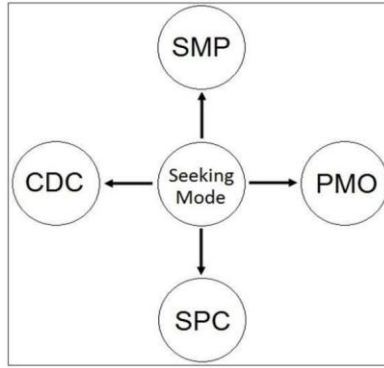


Fig. 8-1:CSO Seeking Mode Parameters

- SMP is used to define the size of seeking memory for each cat. SMP indicates the points explored by the cat. This parameter can be different for different cats.
- SRD declares the mutation ratio for the selected dimensions.
- CDC indicates how many dimensions will be varied. For example, if the search space has 5 dimensions and CDC is set to 0.2, then for each cat, four random dimensions out of the five need to be modified and the other one stays the same.
- SPC is a Boolean flag, which decides whether current position of cat will be one of the candidates to move to or not.

The process of seeking mode is describes as follow:

Step 1. Make j copies of the current location of cat k , where $j = \text{SMP}$. If the value of SPC is true, let $j = (\text{SMP}-1)$, then retain the present position as one of the candidates by Eq.(1).

$$j = \begin{cases} \text{SMP}, & \text{SPC} = "true" \\ \text{SMP} - 1, & \text{SPC} = "false" \end{cases} \quad (1)$$

Step 2. For each copy, according to CDC, randomly plus or minus SRD percent the present values and replace the old ones by Eq.(2).

$$\begin{aligned} Xjd_{new} \\ = (1 + rand * SRD)Xjd_{old} \end{aligned} \quad (2)$$

where Xjd_{old} is the current location; Xjd_{new} is the next location; j denotes the number of cats and d represents the dimensions; and $rand$ is a random value $rand \in [0,1]$.

Step 3. Calculate the fitness values (FS) of all candidate points.

Step 4. If all FS are not exactly equal, calculate the selecting probability of each candidate point by Eq.(3), otherwise set all the selecting probability of each candidate point be 1.

$$p_i = \begin{cases} 1, & \text{when } fS_{max} = fS_{min} \\ \frac{|fS_i - fS_b|}{fS_{max} - fS_{min}}, & \text{when } 0 < i < j, \text{ otherwise} \end{cases} \quad (3)$$

Step 5. Randomly pick the point to move to from the candidate points, and replace the position of cat k.

2.2. Tracing Mode

this mode copies the tracing behaviour of cats. For the first iteration, random velocity values are given to all dimensions of a cat's position. However, for later steps, the next move of each cat is determined based on the velocity of the cat and the best position found by members of cat swarm (Selvakumar et al., 2017). This mode can be summarized in 3 steps as follows:

Step 1. Update velocities ($v_{k,d}$) for all dimensions by Eq.(4).

$$v_{k,d} = v_{k,d} + r_1 c_1 * (x_{best,d} - x_{k,d}) \quad (4)$$

Where, $x_{best,d}$ is the location of the cat with the best fitness value; $v_{k,d}$ is the velocities, $x_{k,d}$ is the current location of cat k in d^{th} dimension. c_1 is a constant value $\in [0,2]$ and r_1 is a uniform random value $\in [0,1]$.

Step 2. Check if the velocities are within the bounds of velocity. In case the new velocity falls out of the range, set it to the limits.

Step 3. Update the position of cat k according to Eq.(5).

$$x_{k,d} = x_{k,d} + v_{k,d} \quad (5)$$

For combining Seeking and Tracing modes the algorithm defines a mixture ratio (MR) which indicates the rate of mixing of seeking mode and tracing mode, as shown in Fig. 8-2 (Selvakumar et al., 2017). MR parameter decides how many cats will be moved into seeking mode process by Eq.(6).

$$C_{seeking} = C_{num} * MR \quad (6)$$

Where $C_{seeking}$ is the number of cuts in the seeking mode, C_{num} is the total number of cats, and MR parameter $\in [0,1]$.

For example, if the population size is 25 and the MR parameter is equal to 0.6, there should be $25 \times 0.6 = 15$ cats move to seeking mode and 10 remaining cats move to tracing mode in this iteration (Orouskhani et al., 2011). We summarized the CSO algorithm below flowchart Fig. 8-4:

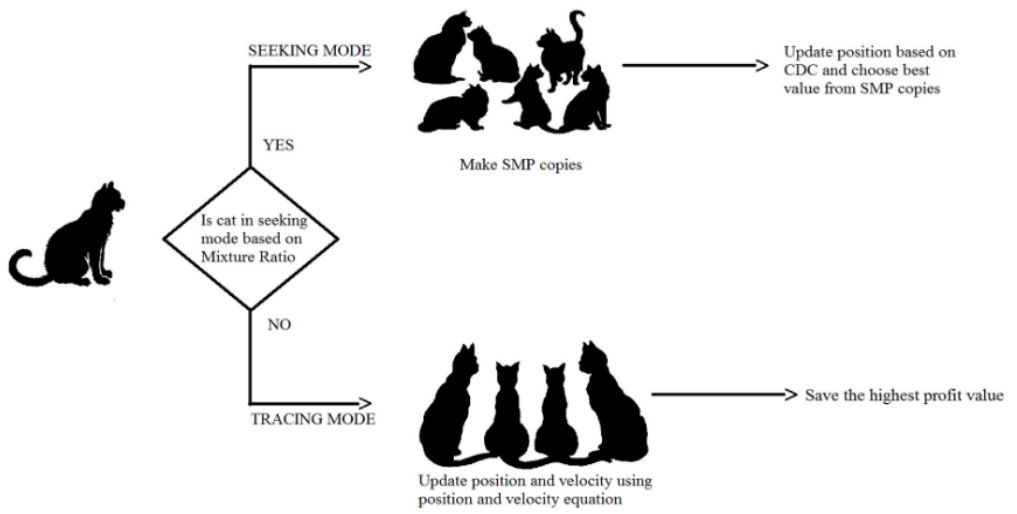


Fig. 8-2:CSO cats mode

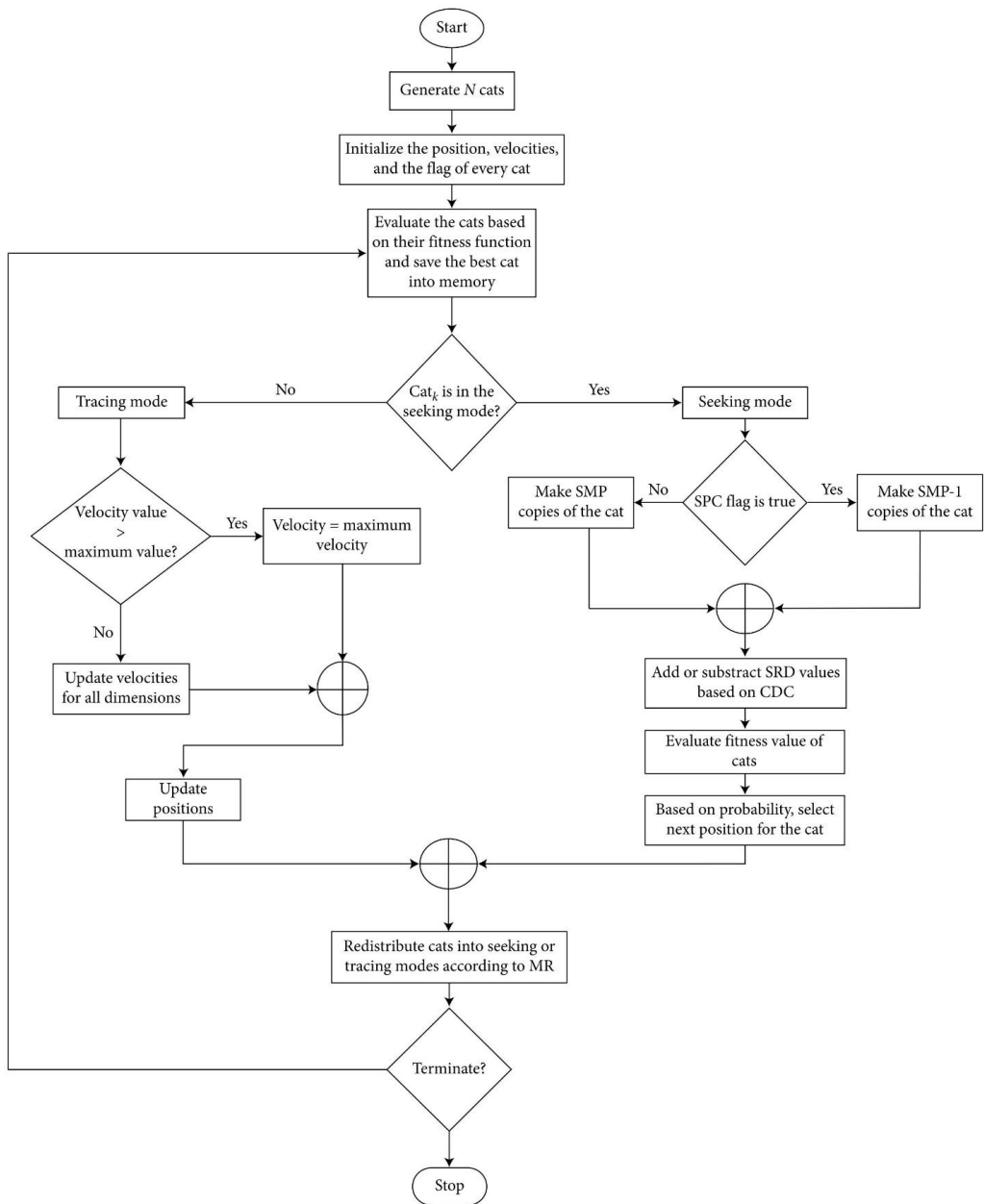


Fig. 8-3: CSO Flowchart (Chu, Tsai, & Pan, 2006)

3. Modifications of CSO

After the appearance of Cat Swarm Optimization (CSO) algorithm in 2006 (Almufti, 2019), many modifications has been applied to the original CSO to improve the performances of the proposed algorithm. In this section, some of the modifications and improvements of the CSO algorithm are listed and arranged according to the development year, as shown in Table 8-1.

Table 8-1: CSO Modifications

ID	ABBR.	NAME	YEAR	AUTHORS	REF.
1.	PCSO	Parallel Cat Swarm Optimization	2008	Tsai et al.	(Tsai et al., 2008)
2.	CSO Clustering	CSO Clustering	2009	Santosa et al.	(Santosa & Ningrum, 2009)
3.	AICSO	Average-Inertia Weighted CSO	2011	Orouskhani et al.	(Orouskhani et al., 2011)
4.	hybrid PCSOABC	hybrid system by combining PCSO with ABC algorithms	2011	Tsai et al.	(Tsai et al., 2011)
5.	CSO + SVM	hybrid system based on SVM and CSO	2011	Wang et al.	(Wang & Wu, 2011)
6.	CSO/PSO + ANN	CSO and PSO algorithms to train ANN	2011	Chittineni et al.	(Chittineni et al., 2011)
7.	EPCSO	Enhanced Parallel Cat Swarm Optimization	2012	Tsai et al.	(Tsai et al., 2012)
8.	MOC SO	Multiobjective Cat Swarm Optimization	2012	Pradhan et al.	(Pradhan & Panda, 2012)
9.	CSO + ANN + OBD	ANN with CSO algorithm and optimal brain damage (OBD) approach	2012	Yusiong	(Yusiong, 2012)
10.	BCSO	Discrete Binary Cat Swarm Optimization Algorithm	2013	Sharafi et al.	(Sharafi et al., 2013)
11.	ADCSO	Adaptive Dynamic Cat Swarm Optimization	2013	Orouskhani et al.	(Orouskhani et al., 2013 a)
12.	ADCSO + GD + ANFIS	combined ADCSO algorithm with gradient descent (GD) algorithm	2013	Orouskhani et al	(Orouskhani et al., 2013b)
13.	Enhanced HCSO	Enhanced Hybrid Cat Swarm Optimization	2014	Hadi et al.	(Hadi & Sabah, 2014)
14.	BCSO + SVM	classification model based on BCSO and SVM	2014	Mohamadeen et al	(Mohamadeen et al., 2014)
15.	ICSO	Improvement Structure of Cat Swarm Optimization	2015	Hadi et al.	(Hadi & Sabah, 2015)
16.	ICSO	Improved Cat Swarm Optimization	2015	Kanwar et al.	(Kanwar et al., 2015)
17.	CSO-GA-PSOSVM	CSO with particle swarm intelligence (PSO), genetic algorithm (GA), and support vector machine (SVM)	2015	Vivek et al.	(Vivek & Reddy, 2015)

18.	MCSO	Modified Cat Swarm Optimization	2015	Lin et al.	(Lin et al., 2015)
19.	CSO + WNN	hybrid system by combining wavelet neural network (WNN) and CSO algorithm	2015	Nanda	(Nanda, 2015)
20.	CCSO + ANN	CSO and ANN that can handle randomness, fuzziness, and accumulative time effect in time series concurrently	2015	Wang et al.	(Wang et al., 2015)
21.	NMCSO	Normal Mutation Strategy-Based Cat Swarm Optimization	2016	Mohapatra et al.	(Mohapatra et al., 2016)
22.	CS-FLANN	combined the CSO algorithm with functional link artificial neural network (FLANN)	2016	Kumar et al.	(Kumar et al., 2016)
23.	OL-ICSO	Opposition-Based Learning-Improved CSO	2017	Kumar et al.	(Kumar & Sahoo, 2017)
24.	CQCSO	Chaos Quantum-Behaved Cat Swarm Optimization	2017	Nie et al.	(Nie et al., 2017)
25.	Hybrid CSO-Based Algorithm	Hybrid CSO-Based Algorithm	2017	Skoullis et al.	(Skoullis et al., 2017)
26.	Hybrid CSO-GA-SA	hybrid system by combining CSO, GA, and SA	2017	Sarswat et al.	(Sarswat et al., 2017)
27.	CSO-CS	Hybrid Cat Swarm Optimization-Crow Search	2017	Pratiwi et al.	(Pratiwi, 2017)
28.	CCSO	Compact Cat Swarm Optimization	2018	Zhao	(Zhao, 2018)
29.	BBCSO	Boolean Binary Cat Swarm Optimization	2018	Siqueira et al.	(Siqueira et al., 2018)

4. Applications:

Since its introduction, CSO and its modified algorithms has potential applied in a wide to solve various problems in several fields. This section presents a review of some of the most important problems, See table 8-2.

Table 8-2: CSO Applications

Algorithm	Domain	Application	Citation
CSO (Original)	Optimization	General global optimization	(Chu et al., 2006)

Parallel CSO (PCSO)	Parallel Computing	Accelerated optimization through parallel processing	(Tsai et al., 2008)
BCSO	Binary Classification	Transformer reliability classification	(Sharafi et al., 2013)
CSO + SVM	Machine Learning	Emotion recognition in e-learning	(Wang & Wu, 2011)
CPNN-CSO	Power Systems / AI	Household electricity consumption forecasting	(Wang et al., 2015)
ADCSO + GD + ANFIS	Hybrid Optimization	Training adaptive fuzzy systems with gradient descent	(Orouskhani et al., 2013)
MOCSO	Multiobjective Optimization	Engineering design with multiple criteria	(Pradhan & Panda, 2012)
CSO-GA-PSO-SVM	AI / Facial Recognition	Facial emotion detection using ensemble bioinspired models	(Vivek & Reddy, 2015)
CQCSO	Renewable Energy	MPPT for photovoltaic systems	(Nie et al., 2017)
ICSO	Smart Grids	DSTATCOM and DGs optimal allocation	(Kanwar et al., 2015)
OL-ICSO	Clustering / Data Mining	Data clustering with opposition-based learning	(Kumar & Sahoo, 2017)
WNN-CSO	Time Series Forecasting	Forecasting chaotic time series	(Nanda, 2015)
Hybrid CSO-CS	Vehicle Routing	Vehicle routing with time windows	(Pratiwi, 2017)

5. Discussions

Cat swarm optimization (CSO) is a Swarm Intelligent optimization algorithm proposed in 2006 by Chu et al.. After its appearance, many modifications has been proposed on it and it has been adapt to solve various problem in different fields. This chapter firstly addressed this overviewed the original CSO algorithm and then it presented some of its modifications, finally some of its applications has been listed.

6. References

- Almufti, S. (2017). Using swarm intelligence for solving NP-hard problems. *Academic Journal of Nawroz University*, 6(3), 46–50. <https://doi.org/10.25007/ajnu.v6n3a78>
- Chu, S., Tsai, P., & Pan, J. (2006). Cat swarm optimization. In *Lecture Notes in Computer Science* (Vol. 4099, pp. 854–858). Springer. https://doi.org/10.1007/11801603_94
- Zebari, A. Y., Almufti, S., & Abdulrahman, C. (2020). Bat algorithm (BA): Review, applications and modifications. *International Journal of Scientific World*, 8(1), 1. <https://doi.org/10.14419/ijsw.v8i1.30120>
- Almufti, S., & Shaban, A. (2018). U-turning ant colony algorithm for solving symmetric traveling salesman problem. *Academic Journal of Nawroz University*, 7(4), 45–49. <https://doi.org/10.25007/ajnu.v6n4a270>
- Almufti, S., Marqas, R., & Asaad, R. (2019). Comparative study between elephant herding optimization (EHO) and U-turning ant colony optimization (U-TACO) in solving symmetric traveling salesman problem (STSP). *Journal of Advanced Computer Science & Technology*, 8(2), 32. <https://doi.org/10.14419/jacst.v8i2.29403>

- Agarwal, P., & Mehta, S. (2014). Nature-inspired algorithms: State-of-art, problems and prospects. *International Journal of Computer Applications*, 100(14), 14–21. <https://doi.org/10.5120/17593-8331>
- Ihsan, R. R., Almufti, S. M., Ormani, B. M. S., Asaad, R. R., & Marqas, R. B. (2021). A survey on cat swarm optimization algorithm. *Asian Journal of Research in Computer Science*, 10(2), 22–32.
- Rajpurohit, J., Sharma, T. K., Abraham, A., & Vaishali. (2017). Glossary of metaheuristic algorithms. *International Journal of Computer Information Systems and Industrial Management Applications*, 9, 181–205.
- Chu, S.-C., & Tsai, P.-W. (2007). Computational intelligence based on the behaviour of cats. *International Journal of Innovative Computing, Information and Control*, 3(1), 163–173.
- Sharafi, Y., Khanesar, M. A., & Teshnehlab, M. (2013). Discrete binary cat swarm optimization algorithm. In *2013 3rd IEEE International Conference on Computer, Control and Communication (IC4)* (pp. 1–6). IEEE. <https://doi.org/10.1109/IC4.2013.6653754>
- Orouskhani, M., Orouskhani, Y., Mansouri, M., & Teshnehlab, M. (2013). A novel cat swarm optimization algorithm for unconstrained optimization problems. *International Journal of Information Technology and Computer Science*, 5(11), 32–41. <https://doi.org/10.5815/ijitcs.2013.11.04>
- Orouskhani, M., Mansouri, M., & Teshnehlab, M. (2011). Average-inertia weighted cat swarm optimization. In *Lecture Notes in Computer Science* (Vol. 6683, pp. 321–328). Springer. https://doi.org/10.1007/978-3-642-21515-5_38
- Selvakumar, K., Vijayakumar, K., & Boopathi, C. (2017). CSO-based solution for load kickback effect in deregulated power systems. *Applied Sciences*, 7(11), 1127. <https://doi.org/10.3390/app7111127>
- Tsai, P. W., Pan, J. S., Chen, S. M., Liao, B. Y., & Hao, S. P. (2008). Parallel cat swarm optimization. In *2008 International Conference on Machine Learning and Cybernetics* (Vol. 6, pp. 3328–3333). IEEE.
- Santosa, B., & Ningrum, M. K. (2009). Cat swarm optimization for clustering. In *2009 International Conference of Soft Computing and Pattern Recognition* (pp. 54–59). IEEE.
- Tsai, P. W., Pan, J. S., Shi, P., & Liao, B. Y. (2011). A new framework for optimization based on hybrid swarm intelligence. In *Handbook of Swarm Intelligence* (pp. 421–449). Springer.
- Wang, W., & Wu, J. (2011). Notice of retraction: Emotion recognition based on CSO & SVM in e-learning. In *Seventh International Conference on Natural Computation* (Vol. 1, pp. 566–570). IEEE.
- Chittineni, S., Mounica, V., Abhilash, K., Satapathy, S. C., & Reddy, P. P. (2011). A comparative study of CSO and PSO trained artificial neural network for stock market prediction. In *International Conference on Computational Science, Engineering and Information Technology* (pp. 186–195). Springer.
- Tsai, P.-W., Pan, J.-S., Chen, S.-M., & Liao, B.-Y. (2012). Enhanced parallel cat swarm optimization based on the Taguchi method. *Expert Systems with Applications*, 39(7), 6309–6319.
- Pradhan, P. M., & Panda, G. (2012). Solving multiobjective problems using cat swarm optimization. *Expert Systems with Applications*, 39(3), 2956–2964.

- Yusiong, J. P. T. (2012). Optimizing artificial neural networks using cat swarm optimization algorithm. *International Journal of Intelligent Systems and Applications*, 5(1), 69–80.
- Sharafi, Y., Khanesar, M. A., & Teshnehlab, M. (2013). Discrete binary cat swarm optimization algorithm. In *3rd International Conference on Computer, Control & Communication (IC4)* (pp. 1–6). IEEE.
- Orouskhani, M., Orouskhani, Y., Mansouri, M., & Teshnehlab, M. (2013). A novel cat swarm optimization algorithm for unconstrained optimization problems. *International Journal of Information Technology and Computer Science*, 5(11), 32–41.
- Orouskhani, M., Mansouri, M., Orouskhani, Y., & Teshnehlab, M. (2013). A hybrid method of modified cat swarm optimization and gradient descent algorithm for training ANFIS. *International Journal of Computational Intelligence and Applications*, 12(2), 1350007.
- Hadi, I., & Sabah, M. (2014). Enhanced hybrid cat swarm optimization based on fitness approximation method for efficient motion estimation. *International Journal of Hybrid Information Technology*, 7(6), 345–364.
- Mohamadeen, K. I., Sharkawy, R. M., & Salama, M. M. (2014). Binary cat swarm optimization versus binary particle swarm optimization for transformer health index determination. In *2014 International Conference on Engineering and Technology (ICET)* (pp. 1–5). IEEE.
- Hadi, I., & Sabah, M. (2015). Improvement cat swarm optimization for efficient motion estimation. *International Journal of Hybrid Information Technology*, 8(1), 279–294.
- Kanwar, N., Gupta, N., Niazi, K. R., & Swarnkar, A. (2015). Improved cat swarm optimization for simultaneous allocation of DSTATCOM and DGs in distribution systems. *Journal of Renewable Energy*, 2015, Article 189080. <https://doi.org/10.1155/2015/189080>
- Vivek, T. V., & Reddy, G. R. (2015). A hybrid bioinspired algorithm for facial emotion recognition using CSO-GA-PSOSVM. In *2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 472–477). IEEE.
- Lin, K. C., Huang, Y. H., Hung, J. C., & Lin, Y. T. (2015). Feature selection and parameter optimization of support vector machines based on modified cat swarm optimization. *International Journal of Distributed Sensor Networks*, 11(7), Article 365869.
- Nanda, S. J. (2015). A WNN-CSO model for accurate forecasting of chaotic and nonlinear time series. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)* (pp. 1–5). IEEE.
- Wang, B., Xu, S., Yu, X., & Li, P. (2015). Time series forecasting based on cloud process neural network. *International Journal of Computational Intelligence Systems*, 8(5), 992–1003.
- Mohapatra, P., Chakravarty, S., & Dash, P. K. (2016). Microarray medical data classification using kernel ridge regression and modified cat swarm optimization-based gene selection system. *Swarm and Evolutionary Computation*, 28, 144–160.
- Kumar, M., Mishra, S. K., & Sahu, S. S. (2016). Cat swarm optimization based functional link artificial neural network filter for Gaussian noise removal from computed tomography images. *Applied Computational Intelligence and Soft Computing*, 2016, Article ID 6304915. <https://doi.org/10.1155/2016/6304915>
- Kumar, Y., & Sahoo, G. (2017). An improved cat swarm optimization algorithm based on opposition-based learning and Cauchy operator for clustering. *Journal of Information Processing Systems*, 13(4), 1000–1013.

- Nie, X., Wang, W., & Nie, H. (2017). Chaos quantum-behaved cat swarm optimization algorithm and its application in the PV MPPT. *Computational Intelligence and Neuroscience*, 2017, Article ID 1583847. <https://doi.org/10.1155/2017/1583847>
- Skoullis, V. I., Tassopoulos, I. X., & Beligiannis, G. N. (2017). Solving the high school timetabling problem using a hybrid cat swarm optimization-based algorithm. *Applied Soft Computing*, 52, 277–289.
- Sarswat, A., Jami, V., & Guddeti, R. M. (2017). A novel two-step approach for overlapping community detection in social networks. *Social Network Analysis and Mining*, 7(1), 47.
- Pratiwi, A. B. (2017). A hybrid cat swarm optimization-crow search algorithm for vehicle routing problem with time windows. In *2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)* (pp. 364–368). IEEE.
- Zhao, M. (2018). A novel compact cat swarm optimization based on differential method. *Enterprise Information Systems*, 2018, 1–25. <https://doi.org/10.1080/17517575.2018.1545984>
- Siqueira, H., Figueiredo, E., Macedo, M., Santana, C. J., Bastos-Filho, C. J., & Gokhale, A. A. (2018). Boolean binary cat swarm optimization algorithm. In *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)* (pp. 1–6). IEEE.
- Chu, S., Tsai, P., & Pan, J. (2006). *Cat swarm optimization*. In *Lecture Notes in Computer Science* (Vol. 4099, pp. 854–858). Springer. https://doi.org/10.1007/11801603_94
- Tsai, P.-W., Pan, J.-S., Chen, S.-M., Liao, B.-Y., & Hao, S.-P. (2008). *Parallel cat swarm optimization*. In *2008 International Conference on Machine Learning and Cybernetics* (Vol. 6, pp. 3328–3333). IEEE.
- Sharafi, Y., Khanesar, M. A., & Teshnehlal, M. (2013). *Discrete binary cat swarm optimization algorithm*. In *2013 3rd IEEE International Conference on Computer, Control and Communication (IC4)* (pp. 1–6). IEEE. <https://doi.org/10.1109/IC4.2013.6653754>
- Wang, W., & Wu, J. (2011). *Emotion recognition based on CSO & SVM in e-learning*. In *Seventh International Conference on Natural Computation* (Vol. 1, pp. 566–570). IEEE.
- Wang, B., Xu, S., Yu, X., & Li, P. (2015). *Time series forecasting based on cloud process neural network*. *International Journal of Computational Intelligence Systems*, 8(5), 992–1003.
- Orouskhani, M., Mansouri, M., Orouskhani, Y., & Teshnehlal, M. (2013). A hybrid method of modified cat swarm optimization and gradient descent algorithm for training ANFIS. *International Journal of Computational Intelligence and Applications*, 12(2), 1350007.
- Pradhan, P. M., & Panda, G. (2012). Solving multiobjective problems using cat swarm optimization. *Expert Systems with Applications*, 39(3), 2956–2964.
- Vivek, T. V., & Reddy, G. R. (2015). A hybrid bioinspired algorithm for facial emotion recognition using CSO-GA-PSOSVM. In *2015 Fifth International Conference on Communication Systems and Network Technologies* (pp. 472–477). IEEE.
- Nie, X., Wang, W., & Nie, H. (2017). Chaos quantum-behaved cat swarm optimization algorithm and its application in the PV MPPT. *Computational Intelligence and Neuroscience*, 2017, Article ID 1583847. <https://doi.org/10.1155/2017/1583847>
- Kanwar, N., Gupta, N., Niazi, K. R., & Swarnkar, A. (2015). Improved cat swarm optimization for simultaneous allocation of DSTATCOM and DGs in distribution systems. *Journal of Renewable Energy*, 2015, Article ID 189080. <https://doi.org/10.1155/2015/189080>

- Kumar, Y., & Sahoo, G. (2017). *An improved cat swarm optimization algorithm based on opposition-based learning and Cauchy operator for clustering*. *Journal of Information Processing Systems*, 13(4), 1000–1013.
- Nanda, S. J. (2015). *A WNN-CSO model for accurate forecasting of chaotic and nonlinear time series*. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)* (pp. 1–5). IEEE.
- Pratiwi, A. B. (2017). *A hybrid cat swarm optimization-crow search algorithm for vehicle routing problem with time windows*. In *2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE)* (pp. 364–368). IEEE

Chapter 9: Bat Algorithm

1. Introduction

The increasing complexity of real world problems causes the researchers to search for efficient techniques. Divide and conquer techniques are efficient way to solve large and complex problems which has been a practice in research since long time such as (NP-hard problems).

Modern optimization algorithms are often nature-inspired, typically based on swarm intelligence. The ways for inspiration are diverse and consequently algorithms can be many deferent types. However, all these algorithms tend to use some specific characteristics for formulating the key updating formulae (Almufti, 2017).

Swarms such as bee colonies, ant colonies, mosquito swarms, fish schools, Bat swarm, have relatively simple behaviours individually, but with amazing capability of co-operations and organizing their actions, they represent a complex and highly structured social organization. Swarm Intelligence (SI) is the field of studying and designing well-organized computational intelligent interactive multi-agent systems that cooperate together to achieve a specific goal and to solving complex optimizations problems by using the behaviour of real living swarms such as birds, fish, bats, and ants (*Almufti et al, 2019*). SI is a part of Artificial Intelligence (AI) introduced by Jing Wang and Gerardo Beni in 1989 in the global optimization framework as a collection of algorithms for controlling robotic swarms (Almufti, 2015)

Bat algorithm (BA) was introduced by Yang in 2010 (zebari et al., 2020). It simulates the echolocation behaviour of microbats as microbats can generate high echolocation. The Bat produces a very high sound to detect its prey which echoes back with some frequency. Echolocation is a process of detecting an object by reflected sound. It is used to know how far the prey is from background object. By observing the bounced frequency of sound, bats are able to distinguish between the prey and obstacle and can sense the distance between them in their nearby surroundings. They fly randomly with some velocity, frequency and sound (loudness) to search for food. Solution of objective function is to find prey at minimum distance. The frequency and zooming parameters

maintain the balance between exploration and exploitation processes. The algorithm continued till convergence criteria are satisfied (Yang & Gandomi, 2012).

2. Bat Algorithm (BA)

Bats are eye-catching animals and their higher potential of echolocation has engrossed interest of scholars from various arenas. Echolocation mechanism is a kind of sonar: bats, mainly micro-bats, create a loud and short pulse of sound and figure out the distance of an object by using the echo reruns back to their ears (zebari et al., 2020). This remarkable positioning method makes bats being able to decide the difference between an obstacle and a prey, allowing them to hunt even in whole darkness (Yang & Gandomi, 2012).

Motivated by the conduct of the bats, Xin-She Yang has developed the Bat Algorithm. Bat algorithm (BA) is a population based metaheuristic algorithm proposed by Yang in 2010 for solving continuous optimization problems (Zebari et al., 2020). The basic BA algorithm is bio-inspired on the bio-sonar or echolocation characteristics of bats. In nature, bats release ultrasonic waves to the environment around it for the purposes of hunting or navigation. After the emission of these waves, it receives the echoes of the waves, and based on the received echo they locate themselves and identify obstacles in their ways and preys as shown in fig.9-1. Furthermore, each agent in swarm is capable of finding the most “nutritious” areas or moving towards a previous best location found by the swarm. Bat algorithm has showed grate efficiency in finding solution in continuous optimization problems (Bora et al., 2012).

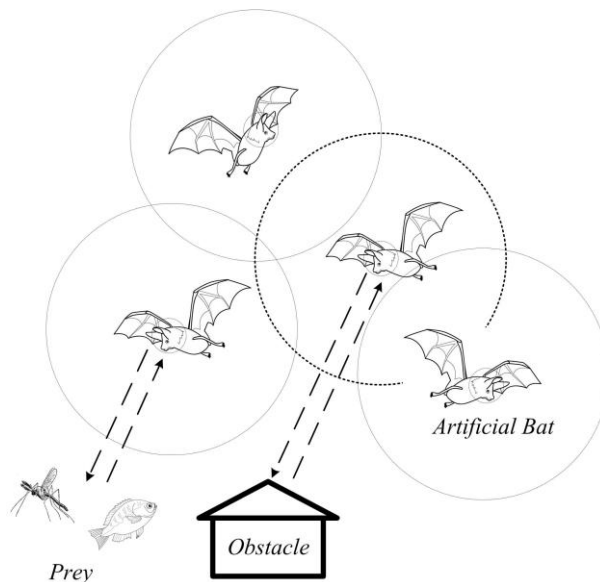


Fig. 9-1: Bat behaviour

In the Bat algorithm (BA), first initialize the bat population then we have to define the pulse frequency, after that we initialize pulse rates and loudness in which we define maximum no of iterations, if result is better than new values will generate and values will updated in velocities. In this random values will generate if solution is yes then we have to select the best solution ,if not then program will move back .when new solutions form in random values it accepts it and find the best current value and output is form(Agarwal & Mehta, 2014). Fig. 9-2 shows the Flowchart of BA.

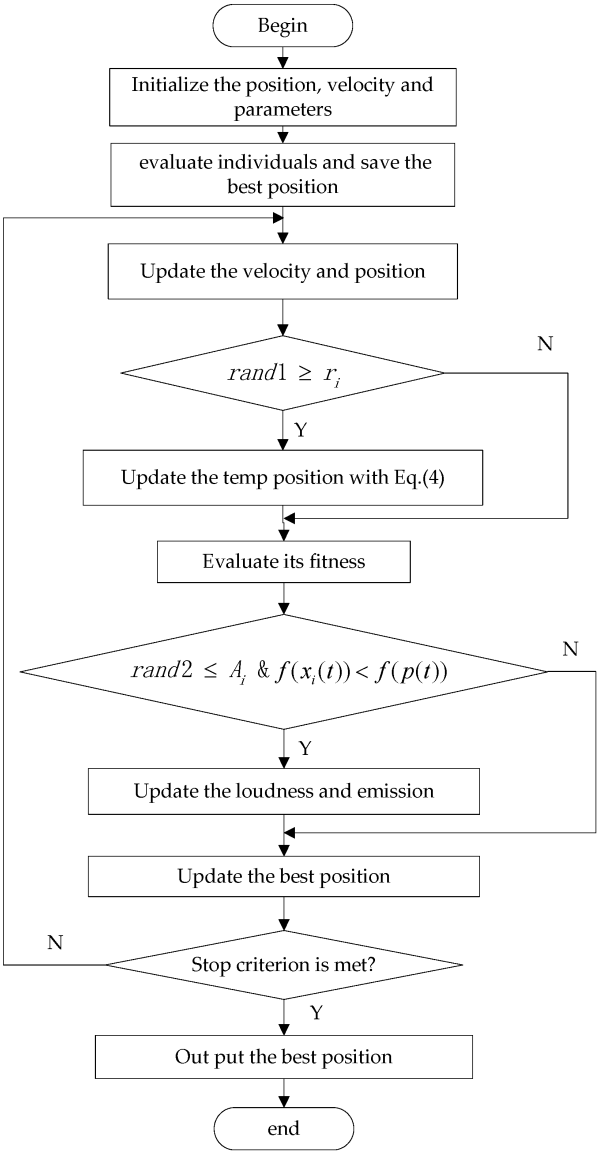


Fig. 9-2: BA Flowchart

BA pseudo code is shown below in (Algorithm 1):

Algorithm 1: Pseudocode of the basic BA:

```

Define the objective function f(x),
Initialize the bat population X = x1, x2, ..., xn,
for each bat xi in the population do Initialize the pulse rate ri, velocity vi and loudness Ai,
Define the pulse frequency fi at xi,
end
repeat
for each bat xi in the population do
Generate new solutions through Equations 1, 2 and 3,
if rand>ri then
Select one solution among the best ones,
Generate a local solution around the best one,
end
if rand<Ai and f(xi)<f(x_) then
Accept the new solution, Increase ri and reduce Ai,
end
end
until termination criterion not reached,
Rank the bats and return the current best bat of the population,
Equations for Generating new solutions

```

First of all, the starting position , velocity , and frequency are initialized for every bat . For each time step t, being T the limit of iterations, the movement of the virtual bats is specified by updating their velocity and position by means of equations 1, 2, and 3 as follows (*Bora et al., 2012*).

$$f_i = f_{min} + (f_{min} - f_{max})\beta \quad (1)$$

$$v_i^t = v_i^{t-1} + [x_i^{t-1} - x_*]f_i \quad (2)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (3)$$

Where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution, f_i denotes frequency of each bat, here , X_* is the current global best solution (location) which is located after comparing all the solutions among all n bats, at each iteration. After the position updating of bats, a random number is generated, if the random number is larger than the pulse emission rate r_i ,a new position will be generated around the current best solutions, and it can be represented by equation (4) (*zebari et al., 2020*).

$$x_{new} = x_{old} + \varepsilon A^t \quad (4)$$

Where, $\varepsilon \in [-1, 1]$, is a random number, while A^t is the average loudness of all the bats at current iteration. Furthermore, the loudness A^t and the pulse emission rate r_i will be updated and a solution will be accepted if a random number is less than loudness A^i and $f(x_i) < f(x_*)$. A^i and r_i are updated by (5).

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0 [1 - \exp(-\gamma t)] \quad (5)$$

Where α and γ are constants, The algorithm iterates until the termination criteria is met.

3. Application

Since the original bat algorithm (BA) has been developed by Yang in 2010 (zebari et al., 2020), bat algorithms and its modifications have been applied in almost every area of optimisation, classifications, image processing, feature selection, scheduling, data mining and others. Table 9-1, will highlight some of the applications of BA (zebari et al., 2020).

Table 9-1: BA applied area

Area/Applications	Author	References
Travelling salesman problem	Li	(Li, 2010)
classification, wireless sensor, data mining	Yang et al,	(Yang, 2010)
solve multi-objective problems	Yang et al,	(Yang & Gandomi, 2012)
brushless direct current (DC) wheel motor problem	Bora et al,	(Bora et al., 2012)
robust turning of power system stabilizer for small signal stability enhancement	Sambariya et al,	(Sambariya & Prasad, 2014)
load frequency control	Sathya et al,	(Sathya & Ansari, 2015)
node localization of wireless sensor networks based	Sun et al,	(Sun & Xu, 2015)
low energy adaptive clustering hierarchy protocol based	Cao et al,	(Cao et al., 2014)
applications in big data and machine learning	Cui et al,	(Cui et al., 2018)
support vector data description(SVDD)(CBA-SVDD)to design effective descriptions of data facial feature selection	Hamidzadeh et al, Alsalibi et al	(Hamidzadeh et al., 2017) (Alsalibi et al., 2017)
vector machine (SVM) parameters that reduce the classification error	Yang et al,	[24]

biomedical research	Tharwat et al.	(Tharwat et al., 2016)
Fuel arrangement optimization of reactor core	Kashi et al.	(Kashi et al., 2014)
Multilevel image thresholding	Alihodzic and Tuba	(Alihodzic & Tuba, 2014)
Economic dispatch	Latif and Palensky	(Latif & Palensky, 2014)
Feature selection	Taha et al.	(Taha et al., 2013)
Planning the sport sessions	Fister et al.	(Fister et al., 2014)
Application in Multiple UCAVs	Li and Peng	(Li & Peng, 2014)
Design of a conventional power system stabilizer	Sambariya and Prasad	(Sambariya & Prasad, 2014)
Toy model of protein folding	Cai et al.	(Cai et al., 2014)
Design of skeletal structures	Kaveh and Zakian	(Kaveh & Zakian, 2014)
face recognition	Alslibi	(Alslibi et al., 2017)

4. Modification to Bat algorithm (BA)

To improve the enactment of BA, many modifications and strategies have been tried, and many new variants of the bat algorithm are created in the recent literature. Table 9-2, summarize some the latest variants of bat algorithm.

Table 9-2: Bat algorithm Variants Modifications

Modifications on BA	Authors	References	Descriptions
fuzzy bat algorithm	Nikov et al,	(Nikov et al., 2011)	The modified algorithm was experienced on cluster analysis and found to be very effective
binary bat algorithm	Nakamura et al,	(Nakamura et al., 2012)	A binary bat algorithm for solving the well known economic load dispatch problem with the valve-point effect
binary bat algorithm that	Sabba et al,	(Sabba & Chikhi, 2014)	With the help of sigmoid function only binary allowed to new bat's position.

uses the sigmoid function discrete binary variant of BA	Sabba and Chikhi	(Sabba & Chikhi, 2014)	They used this variant in feature selection problem discrete binary variant of BA and tested it on multidimensional knapsack problem. With experimental results they concluded that this discrete binary BA outperformed the standard BA
bat algorithm with mutation	Zhang and Wang	(Zhang & Wang, 2012)	a new bat algorithm with mutation for image processing. They made two modifications to original BA. First, they used fixed frequency and loudness and second, they added a mutation operator to increase the diversity of the population. They tested it on image processing and found that the proposed algorithm produce good result than standard BA. The standard BA hybridized with differential evolution techniques. This hybridization enhanced the local search capability of original BA.
opposition-based BAT algorithm	Sabba et al	(Sabba & Chikhi, 2014)	Sabba et al. improved the convergence speed of BA by embedding the opposition based numbering concept. They tested it against several benchmark functions. Simulation results showed that their approach increases the accuracy and convergence speed of BA.
BAT algorithm with levy flights trajectory	Xie et al	(Xie et al., 2013)	Xie et al. improved the low accuracy rate and slow convergence speed of BA. They introduced levy flights trajectory which increases the diversity of population, so that the algorithm effectively jump out of local minima. They also used the differential operator to accelerate the convergence speed. The proposed algorithm was tested on typical benchmark functions and they concluded that their approach has superior approximation capabilities in high dimensional space.
Chaotic BA (CBA)	Afrabandpey et al.	(Afrabandpey et al., 2014)	Afrabandpey et al. used chaotic sequence for parameter initialization of BA. They called it Chaotic BA (CBA). They studied the effect of different chaotic sequence on convergence behaviour of BA. Simulation result showed that CBA outperforms the BA.

employed chaos in original BA	Gandomi and Yang	(Gandomi & Yang, 2014)	Gandomi and Yang chaos in original BA. They developed four different chaotic BA variants and used 13 different chaotic maps to validate each variant. They concluded that their approach increase the global search mobility of BA.
Improving the exploration mechanism of original BA	Yilmaz et al	(Yilmaz et al., 2014)	Yilmaz et al improved the exploration mechanism of original BA. They changed the equation of loudness and pulse emission rate of bats. They tested this modified bat algorithm (MBA) on 15 different benchmark functions and concluded that MBA performs better than BA
enhancing the explorative mechanism of BA	Li and Zhou	(Li & Zhou, 2014)	Li and Zhou also enhanced the explorative mechanism of BA by introducing complex value encoding scheme into BA. They update real and imaginary part of complex encoding separately which increases the diversity of population.
bat algorithm with Gaussian walk	Cai et al.	(Cai et al., 2014)	They improved the local search capability by introducing the Gaussian walk instead of uniform random walk. They also changed the velocity update equation of BA which results in high population diversity. This approach expands the search dimensions.
cloud model BA (CBA)	Zhou et al	(Zhou et al., 2014)	They incorporated cloud model concept into BA and called it cloud model BA (CBA). Cloud model has excellent characteristics of representing uncertain knowledge. They remodeled the echolocation model of BA by utilizing the transformation theory of cloud model. They studied that proposed algorithm had good performance on function optimizations
compact bat algorithm (cBA)	Dao et al	(Dao et al., 2014)	compact bat algorithm (cBA) was developed for limited hardware resources environments. They replaced the design variable of solution search space of BA with a probabilistic representation of the population. Their study showed that this approach can be effectively used in limited memory case.

self-adaptive bat algorithm	Fister et al	(Fister et al., 2014)	Fister et al. presented a self-adaptive bat algorithm in which control parameters were self-adapted in the similar way like self-adaptive DE algorithm. They tested it on ten benchmark functions and found that proposed method can be used in continuous optimization efficiently.
hybridized the self adaptive bat algorithm	Fister et al	(Fister et al., 2014)	Fister et al. hybridized the self adaptive bat algorithm with different DE strategies. These techniques improved the local search capability of the proposed algorithm
Enhanced BA	Yilmaz and Küçüksill	(Yilmaz & Küçüksille, 2015)	local and global search capability of BA was improved by using inertia weight modification, distribution of the population modification, and hybridization with invasive weed optimization algorithm
double sub-population levy flight BA (DLBA)	Jun et al.	(Jun et al., 2015)	for enhancing local and global search ability, Jun et al. developed a double sub-population levy flight BA (DLBA) They employed two subgroups namely external subgroup and internal subgroup. Global exploration improved by external subgroup and local exploitation was improved by the internal subgroup. They tested proposed algorithm on several test functions and concluded DLBA can outperform the BA.
Doppler Effect with standard BA.	Meng et al.	(Meng et al., 2015)	Meng et al. introduced the bat's habitat selection and their self-adaptive compensation for Doppler Effect in echoes into the standard BA.
Improved BA (IBA)	Wang et al	(Wang et al., 2016)	Wang et al improved version of BA called it improved BA (IBA). They combine BA with DE in order to select the best solution in the bat population. They used this algorithm in three dimensional path planning problem and concluded that proposed approach can performed better than BA.
greedy randomized adaptive search	Zhou et al.	(Zhou et al., 2016)	Zhou et al. successfully integrated the greedy randomized adaptive search procedure and path relinking into the standard BA. They used it in capacitated

procedure with BA			vehicle routing problem and found it very effective. For solving multi-model numerical problems.
using optimal forage strategy in BA	Cai et al	(Cai et al., 2016)	They improved the local search ability by using optimal forage strategy. They also introduced a random disturbance strategy to enhance the global search ability in multi-model environment.
quantum behaved mean best position BA (QMBA)	Zhu et al	(Zhu et al., 2016)	Zhu et al. proposed a quantum behaved mean best position BA (QMBA) for improving the convergence speed of BA. In early stages of this algorithm, the position of each bat updated by current best solution and in later stages, bat's position depends upon the mean best position.
hybrid multi objective shuffled BA (MOsh-BAT)	Yammani et al.	(Zhu et al., 2016)	proposed a hybrid multi objective shuffled BA (MOsh-BAT). They combine the features of shuffled frog leaping algorithm (SFLA) and BA. The exploration capability of BA and exploitation method of SFLA was combined to form a new optimization algorithm.

5. Conclusion

Based on the behaviour of Bat a Method has been designed by Yang in 2010 called Bat Algorithm (BA) . This algorithm has proved to be better than other nature inspired algorithm. This algorithm has also been applied to many problems such as: classification and data mining, image process and fuzzy logic etc. BA is a very promising technique which can be further expired for application in many areas. Further, its hybrids can also be developed and tested for various engineering problems.

This chapter shows that Bat algorithm (BA) has become a powerful nature inspired metaheuristic algorithm for many continuous and discrete optimization problems. Nowadays, BA has widely expanded its implementation in almost every area of optimization and engineering applications. This chapter provides an updating literature review on applications and modifications of BA

6. References

Almufti, S. (2017). Using swarm intelligence for solving NP-Hard problems. *Academic Journal of Nawroz University*, 6(3), 46–50. <https://doi.org/10.25007/ajnu.v6n3a78>

- Almufti, S., Marqas, R., & Ashqi, V. (2019). Taxonomy of bio-inspired optimization algorithms. *Journal of Advanced Computer Science & Technology*, 8(2), 23. <https://doi.org/10.14419/jacst.v8i2.29402>
- Almufti, S. (2015). *U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem*. <http://hdl.handle.net/11129/1734>
- Agarwal, P., & Mehta, S. (2014). Nature-inspired algorithms: State-of-art, problems and prospects. *International Journal of Computer Applications*, 100(14), 14–21. <https://doi.org/10.5120/17593-8331>
- Li, Y. (2010). Solving TSP by an ACO- and BOA-based hybrid algorithm. In *2010 International Conference on Computer Application and System Modeling* (pp. 189–192). IEEE.
- Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature-Inspired Cooperative Strategies for Optimization* (pp. 65–74). Springer.
- Almufti, S. M. (2019). Historical survey on metaheuristics algorithms. *International Journal of Scientific World*, 7(1), 1. <https://doi.org/10.14419/ijsw.v7i1.29497>
- Zebari, A. Y., Almufti, S. M., & Abdulrahman, C. M. (2020). Bat algorithm (BA): Review, applications and modifications. *International Journal of Scientific World*, 8(1), 1–7. Science Publishing Corporation.
- Yang, X. S., & Gandomi, A. H. (2012). Bat algorithm: A novel approach for global engineering optimization. *Engineering Computations*, 29, 464–483.
- Bora, T. C., Coelho, L. D. S., & Lebensztajn, L. (2012). Bat-inspired optimization approach for the brushless DC wheel motor problem. *IEEE Transactions on Magnetics*, 48, 947–950.
- Sambariya, D. K., & Prasad, R. (2014). Robust tuning of power system stabilizer using metaheuristic bat algorithm. *International Journal of Electrical Power & Energy Systems*, 61, 229–238.
- Sathya, M. R., & Ansari, M. M. T. (2015). Load frequency control using bat inspired algorithm based PI controllers. *International Journal of Electrical Power & Energy Systems*, 64, 365–374.
- Sun, S., & Xu, B. (2015). Node localization of wireless sensor networks based on hybrid bat-quasi-Newton algorithm. *Journal of Computer Applications*, 11, 38–42.
- Cao, Y., Cui, Z., Li, F., Dai, C., & Chen, W. (2014). Improved LEACH protocol based on local centroid bat algorithm. *Sensor Letters*, 12, 1372–1377.
- Cui, Z., Xue, F., Cai, X., Cao, Y., Wang, G. G., & Chen, J. (2018). Detection of malicious code variants based on deep learning. *IEEE Transactions on Industrial Informatics*, 14, 3187–3196.
- Hamidzadeh, J., Sadeghi, R., & Namaei, N. (2017). Weighted SVDD based on chaotic bat algorithm. *Applied Soft Computing*, 60, 540–551.
- Alslibi, B., Venkat, I., & Al-Betar, M. A. (2017). A membrane-inspired bat algorithm to recognize faces. *Engineering Applications of Artificial Intelligence*, 64, 242–260.
- Tharwat, A., Hassanien, A. E., & Elnaghi, B. E. (2016). BA-based algorithm for SVM parameter optimization. *Pattern Recognition Letters*, 93, 13–22.
- Kashi, S., Minuchehr, A., Poursalehi, N., & Zolfaghari, A. (2014). Bat algorithm for fuel arrangement optimization. *Annals of Nuclear Energy*, 64, 144–151.
- Alihodzic, A., & Tuba, M. (2014). Improved bat algorithm for image thresholding. *The Scientific World Journal*, 2014, 1–14.

- Latif, A., & Palensky, P. (2014). Economic dispatch using modified bat algorithm. *Algorithms*, 7(3), 328–338.
- Taha, A. M., Mustapha, A., & Chen, S. D. (2013). Naive Bayes-guided bat algorithm for feature selection. *The Scientific World Journal*.
- Fister, I., Rauter, S., Yang, X. S., & Ljubič, K. (2014). Planning sports training with bat algorithm. *Neurocomputing*.
- Li, Y. G., & Peng, J. P. (2014). An improved bat algorithm and its application in multiple UCAVs. *Applied Mechanics and Materials*, 442, 282–286.
- Sambariya, D., & Prasad, R. (2014). Robust tuning of power system stabilizer for small signal stability using bat algorithm. *International Journal of Electrical Power & Energy Systems*, 61, 229–238.
- Cai, X., Wang, L., Kang, Q., & Wu, Q. (2014). Bat algorithm with Gaussian walk. *International Journal of Bio-Inspired Computation*, 6(3), 166–174.
- Kaveh, A., & Zakian, P. (2014). Enhanced bat algorithm for optimal design of skeletal structures. *Asian Journal of Civil Engineering*, 15(2), 179–212.
- Alsalibi, B., Venkat, I., & Al-Betar, M. (2017). A membrane-inspired bat algorithm for facial recognition. *Engineering Applications of Artificial Intelligence*, 64, 242–260. <https://doi.org/10.1016/j.engappai.2017.06.018>
- Nikov, K., Nikov, A., & Sahai, A. (2011). A fuzzy bat clustering method for ergonomic screening. In *S3T International Conference* (pp. 59–66).
- Nakamura, R., Pereira, L., Costa, K., Rodrigues, D., Papa, J., & Yang, X. (2012). BBA: A binary bat algorithm for feature selection. *XXV SIBGRAPI Conference*, 291–297.
- Sabba, S., & Chikhi, S. (2014). A discrete binary bat algorithm for multidimensional knapsack problem. *International Journal of Bio-Inspired Computation*, 6(2), 140–152.
- Zhang, J., & Wang, G. (2012). Image matching using a bat algorithm with mutation. *Applied Mechanics and Materials*, 203, 65–74.
- Xie, J., Zhou, Y., & Chen, H. (2013). A novel bat algorithm based on differential operator and Lévy flights. *Computational Intelligence and Neuroscience*, 1–13.
- Afrabandpey, H., Ghaffari, M., Mirzaei, A., & Safayani, M. (2014). A novel bat algorithm based on chaos. *Iranian Conference on Intelligent Systems (ICIS)*, 1–6.
- Gandomi, A., & Yang, X. (2014). Chaotic bat algorithm. *Journal of Computational Science*, 5(2), 224–232.
- Yilmaz, S., Kucuksille, E., & Cengiz, Y. (2014). Modified bat algorithm. *Elektronika ir Elektrotechnika*, 20(2), 71–78.
- Li, L., & Zhou, Y. (2014). A novel complex-valued bat algorithm. *Neural Computing and Applications*, 25(6), 1369–1381.
- Cai, X., Wang, L., Kang, Q., & Wu, Q. (2014). Bat algorithm with Gaussian walk. *International Journal of Bio-Inspired Computation*, 6(3), 166–174.
- Zhou, Y., Xie, J., Li, L., & Ma, M. (2014). Cloud model bat algorithm. *The Scientific World Journal*, 1–11.
- Dao, T., Pan, J., Nguyen, T., Chu, S., & Shieh, C. (2014). Compact bat algorithm. In *Intelligent Data Analysis and Its Applications* (Vol. 2, pp. 57–68). Springer.
- Fister, I., Fong, S., Brest, J., & Fister, I. (2014). Toward self-adaptation in the bat algorithm. In *IASTED Conference on Artificial Intelligence and Applications* (pp. 400–406).

- Fister, I., Fong, S., Brest, J., & Fister, I. (2014). A novel hybrid self-adaptive bat algorithm. *The Scientific World Journal*, 1–12.
- Yilmaz, S., & Küçüksille, E. (2015). A new modification on bat algorithm for optimization. *Applied Soft Computing*, 28, 259–275.
- Jun, L., Liheng, L., & Xianyi, W. (2015). A double-subpopulation bat algorithm variant. *Applied Mathematics and Computation*, 263, 361–377.
- Meng, X., Gao, X., Liu, Y., & Zhang, H. (2015). Bat algorithm with habitat selection and Doppler effect. *Expert Systems with Applications*, 42(17–18), 6350–6364.
- Wang, G., Chu, H., & Mirjalili, S. (2016). 3D path planning for UCAV using improved bat algorithm. *Aerospace Science and Technology*, 49, 231–238.
- Zhou, Y., Luo, Q., Xie, J., & Zheng, H. (2016). Hybrid bat algorithm with path relinking for CVRP. In *Metaheuristics and Optimization in Civil Engineering*, 7, 255–276.
- Cai, X., Gao, X., & Xue, Y. (2016). Improved bat algorithm with forage and disturbance strategy. *International Journal of Bio-Inspired Computation*, 8(4), 205–214.
- Zhu, B., Zhu, W., Liu, Z., Duan, Q., & Cao, L. (2016). A quantum-behaved bat algorithm with mean-best guidance. *Computational Intelligence and Neuroscience*, 1–17.

Chapter 10: artificial bee colony Algorithm

1. Introduction

Optimization is essential for addressing numerous real-world challenges across various domains, including engineering design, data mining, machine learning, logistics, finance, and beyond. However, the inherent complexity of these problems often renders traditional optimization methods, such as gradient-based techniques, inadequate in certain scenarios. In response, metaheuristic algorithms, which draw inspiration from natural processes, have emerged as robust alternatives for solving such intricate optimization problems. Swarm intelligence algorithms, in particular, have proven highly effective. These algorithms are inspired by the collective behaviour of decentralized, self-organized systems found in nature.

The Artificial Bee Colony (ABC) algorithm, developed by Derviş Karaboga in 2005 (Almufti et al., 2022), is a prominent swarm intelligence-based optimization technique. It models the foraging behaviour of honeybee colonies, which efficiently locate optimal food sources through a combination of exploration and exploitation. In nature, honeybees communicate the quality of food sources using a "waggle dance," guiding other bees in the colony to promising locations. This behaviour is replicated in the ABC algorithm, where each bee represents a potential solution to the optimization problem, and the colony collaboratively refines these solutions over iterations (Almufti, 2022a)

The ABC algorithm has gained significant popularity due to its simplicity, ease of implementation, and effectiveness in solving both constrained and unconstrained optimization problems. Similar to other nature-inspired algorithms like Water Evaporation Optimization(WEO) (Almufti, 2023a), Particle Swarm Optimization (PSO), Social Spider Optimization Algorithm (SSO) (Almufti, 2021), Artificial fish swarm algorithm(AFS) (Neshat et al., 2014), Ant Colony Optimization (ACO) (Almufti, 2022), Cat Swarm Optimization(CSO) (Ihsan et al., 2021), Big Bang-Big Crunch Algorithm(BB-BC) (Almufti, 2023b), ABC excels in tackling nonlinear, multidimensional, and NP-hard problems, which are common in practical applications.

A distinguishing feature of ABC is its ability to maintain a balance between exploration (searching new areas of the solution space) and exploitation (refining existing solutions). This balance is achieved through the coordinated efforts of three types of bees: employed, onlooker, and scout bees.

2. artificial bee colony (ABC) algorithm

The artificial bee colony in the ABC algorithm is composed of three groups of bees: employed bees, onlookers and scouts. Where a bee waiting in the dance area for making a decision on a food source of choice is called an onlooker(Almufti et al., 2021). On the other hand, a bee visiting the food source previously visited by itself is called an employed bee. While a bee performing random search is named a scout. In the artificial bee colony algorithm, the colony is divided into two halves. The first of which consist of employed bees, while the second consists of onlookers. There is only one bee employed to each and every food source. i.e., the number of employed bees is that of the food source around the bee hive. When the food source of an employed be is exhausted by the bee itself. Then the employed bee and onlooker bees become scout bees. The main steps of the algorithm can be seen below(Karaboga, 2005) (Almufti, 2023a)

- Initialize.
- REPEAT.
 - (a) Place the employed bees on the food sources in the memory;
 - (b) Place the onlooker bees on the food sources in the memory;
 - (c) Send the scouts to the search area for discovering new food sources.
- UNTIL (requirements are met).

Each cycle of search in the ABC algorithm consists of three steps: firstly, an employed bee is sent onto a food source and then their nectar amounts are measured. Secondly, food source selection by the onlooker bees after sharing the information of employed bees regarding their nectar amounts from the food source. Finally, choosing scout bees and sending them to possible food sources(Zhou et al, 2017).

Initially, a set of food sources are randomly selected by the bees and their nectar amount is measured. Afterwards, the bees return to the hive and share the nectar information gathered from the different food sources with the bees waiting in the dance area. On stage two, after the information is shared. All employed bees return to the food source vicinity previously visited by herself since the food source place is fresh in their memory. And so, the bees choose a new food source relying on visual information in the area of the present food source. On stage three, an onlooker decides on a food place depending on the nectar information shared on the dance are by the employed bee. The probability of choosing a food source by the onlooker increases with the increase of the nectar amount of that food source. Hence, employee bees that carry higher amount of nectar recruit the onlooker bees for the higher nectar food source area. After an onlooker arrive

at the chosen place, she attempts to select a food source in the vicinity of the one in memory relying on visual information. A new food source is arbitrarily selected when a food source is abandoned by the bees. The new food source is randomly chosen by a scout to replace the one left behind.

In this model, at most one scout bee roams outside to look for a new food source at each cycle. While the number of employed bees and onlooker bees is equal. In the artificial bee colony algorithm, the possible solution of the optimization is represented by the position of the food source. While the quality or fitness of the solution is represented by the nectar amount of the food source. The number of possible solutions to the optimization problem is equal to the number of employed or onlooker bees.

At the beginning, ABC algorithm produces a randomly distributed initial population $P(G = 0)$ of SN solution, i.e. food source positions, where population size is denoted by SN. Each solution (food source) x_i ($i = 1, 2, \dots, SN$) is a D-dimensional vector. Where D denotes the number of optimization parameters. After the initialization, the population of the solutions (positions) undergoes repeated cycles, $C = 1, 2, \dots, C_{max}$, of the search processes of the employed bees, the onlooker bees and scout bees. An artificial employed or onlooker bee probabilistically modifies the position (solution) in her memory for locating a new food source and tests the nectar amount (fitness value) of the new source (new solution).

Real bees produce new food sources based on a comparing information gathered about food sources in the area around visually. On the other hand, bees in an ABC algorithm produce new food sources based on the comparison of new food sources position. However, the comparison does not occur based on any information but on a modification of the existing source in their memory as described in Equ. (4) provided that the new food source nectar is higher than the one in their memory. So, the bee memorizes the new position and forget the old one. Otherwise, the position of the old food source is kept. After the search process is concluded by the bees, nectar information of the food sources (solutions) and their respective position is shared with onlookers on the dance area. Onlooker bees then evaluate the information received from all the employed bees and select a food source with a probability related to its nectar amount.

Onlooker bees select food sources depending on the probability value of a food source (p_i) given in the expression below (3):

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (3)$$

Where fit_i is the fitness value of a solution i calculated by its artificial employed bee. The fitness value is proportional to the amount of nectar of the food source in position i . while SN denote the number of food sources which is the same as the number of employed or onlooker bees. This way, a candidate food position is produced from the old position through the exchange of information from the employed bees with the onlooker bees using the equation (4)

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \quad (4)$$

where $k \in \{1, 2, \dots, BN\}$ and $j \in \{1, 2, \dots, D\}$ are arbitrarily chosen indices. K must be randomly determined yet different from i . $\phi_{i,j}$ is a random number between $[-1, 1]$. Which controls the production of a neighbor food source position around $x_{i,j}$ while the modification represents the visual comparison of neighboring food source position by the bee. Equation 2.2 indicates that as the difference between $x_{i,j}$ and $x_{k,j}$ is decreased, the perturbation on the position $x_{i,j}$ is also decreased. Thus, as the search edges closer to the optimum solution in the search space, the step length is adaptively reduced. The parameters which exceed their predetermined limit are reset to an acceptable value. In this model, the parameters that exceed their limit are reset to their limit value. The scouts replace abandoned food sources left behind by the bees with new food sources. In the ABC algorithm, this is simulated by randomly introducing new positions and replacing the abandoned ones. While if a position cannot be further improved with a preset number of cycles (limit), then the food source is assumed to be abandoned. A when a candidate source position $v_{i,j}$ is produced, it is evaluated by the bees. If its performance equals or is better than the food source in the memory. Then it is set as the food source of choice. However, if its performance value is lower, then the old food source is retained. This can be described as a greedy selection mechanism.

In artificial bee colony algorithm, there are four distinctive selection processes: (1) a global selection procedure used by the onlookers for finding promising areas as described by (2.1), (2) a local selection procedure done in an area by the employed bees and the onlookers relying on local information (for real bees, this is the colour, shape and fragrance of the flowers) (bees cannot differentiate the type of nectar unless they inspect the source on and differentiate among sources there based on their scent) for finding a neighbor food source around the source in the memory as defined in (2.2), (3) greedy selection process, which is a local selection procedure executed by all bees where a candidate food source is compared to the one in memory, and the one with the higher nectar amount is chosen while the other is forgotten. (4) a random selection process carried out by scouts.

The explanation above clearly indicates that there are three control parameters in the basic ABC: The number of the food sources which is equal to the number of employed or onlooker bees (SN), the limit value and the maximum cycle number (MCN). Regarding honey bees, the recruitment rate indicates a measure of how fast the bee colony locates and exploits a newly found food source (Faris et al., 2020). On the other hand, artificial bees similarly represent the recruitment rate with the speed with which an optimal or high-quality solution of an optimization problem is discovered. In the real world, survival of the fittest indicates that in order for bee colonies to survive they must find and exploit the best food sources rapidly. Meanwhile, the discovery of an optimal solution of engineering problems is related to the rapid discovery of good solution for that problem specifically for problems that must be solved in real time. In a high-quality search process, exploration and exploitation processes must be carried out

simultaneously. In the ABC algorithm, the scouts control the exploration process, while onlookers and employed bees carry out the exploitation process in the search space.

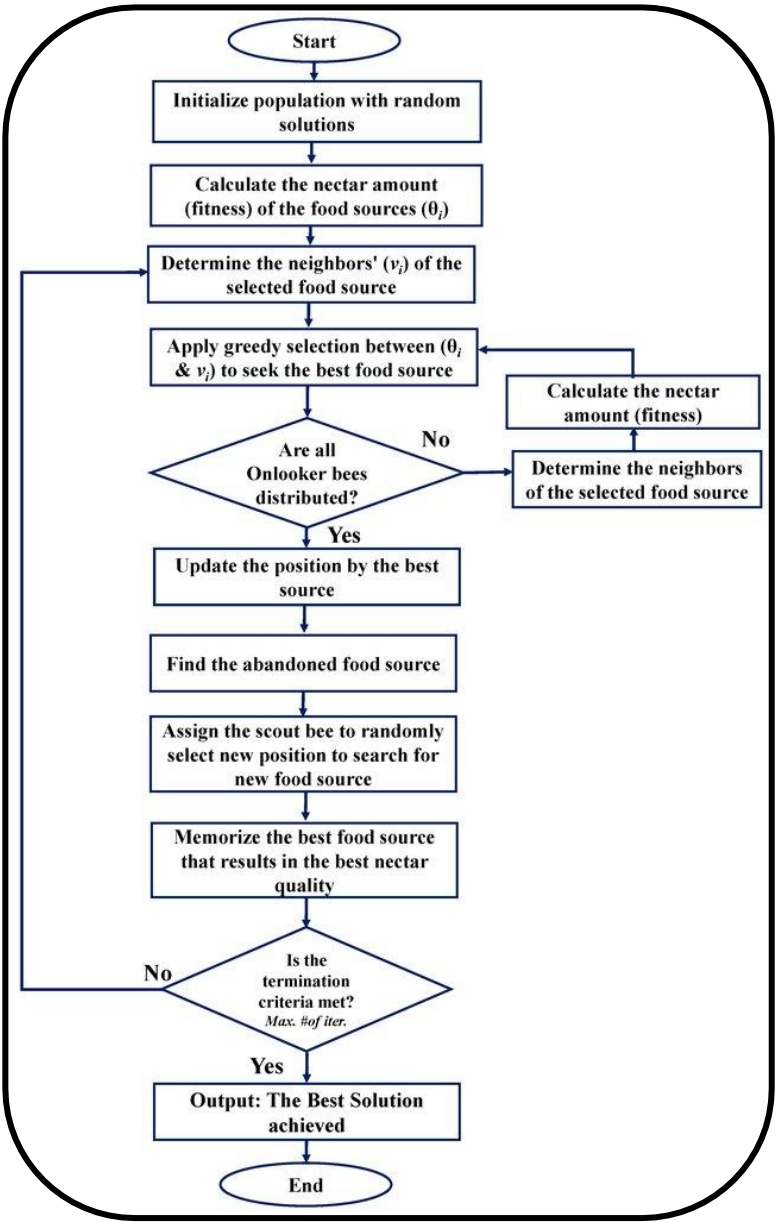


Fig. 10-1: ABC Algorithm Flowchart

Fig. 10-1 illustrates the operational flow of the ABC algorithm. The process initiates with a population of randomly generated candidate solutions, each corresponding to a food source. The quality of each food source is evaluated using a predefined fitness function, analogous to the nectar amount (Mirjalili & Mirjalili, 2021).

Step 1. Initialization

A population of food sources is randomly initialized, where each solution vector represents a potential candidate for the optimization problem. The initial fitness (or nectar amount, denoted as θ_i of each solution is calculated.

Step 2. Employed Bee Phase

Each employed bee explores the vicinity of its associated food source by generating a new solution v_i based on a neighbourhood search operator. A greedy selection mechanism is then applied to retain the solution with higher fitness between θ_i and v_i . This phase allows exploitation around promising areas in the search space (Tharwat & Hassanien, 2022).

Step 3. Onlooker Bee Phase

Onlooker bees probabilistically select food sources based on the fitness values calculated during the previous phase. Each selected source undergoes a similar neighbourhood search and greedy selection. This probabilistic selection mechanism Favors high-quality solutions, thus intensifying the search around them.

A check is performed to ensure that all onlooker bees have been allocated. If not, the fitness evaluation and neighbourhood search are repeated(Singh & Singh, 2010).

Step 4. Scout Bee Phase

Food sources that have not improved over a predetermined number of iterations (termed as the "limit") are considered abandoned. The corresponding employed bee becomes a scout bee and generates a completely new food source by exploring the global search space randomly. This mechanism promotes exploration and helps prevent premature convergence. (Singh & Singh, 2010).

Step 5. Memorization and Termination

The algorithm memorizes the best solution found so far. If the termination criterion—typically the maximum number of iterations or a convergence threshold—is satisfied, the algorithm halts and outputs the best solution discovered. Otherwise, the algorithm loops back to the employed bee phase and continues the iterative process (Sharma & Pant, 2011).

3. Modifications of Artificial Bee Colony Algorithm

Over the years, numerous modifications have been proposed to improve the performance of the ABC algorithm, addressing issues such as slow convergence, local optima entrapment, and poor exploitation in some problem spaces[(Coelho & Alotto, 2012). Table 10-1, shows modifications of ABC.

3.1. Hybrid Approaches

One common modification is the hybridization of ABC with other metaheuristics such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE). These hybrid methods enhance ABC by combining the global search capability of ABC with the rapid convergence of other algorithms. For instance, hybrid ABC-PSO algorithms are used to balance exploration and exploitation more effectively(Singh et al., 2013)

3.2. Dynamic Control Parameters

To improve convergence and adaptability, dynamic control parameters have been introduced, allowing ABC to adjust the number of bees, mutation rates, and neighbourhood size dynamically during the optimization process. This improves the algorithm's ability to explore more effectively in high-dimensional or complex search spaces (Panigrahi & Agrawal, 2014).

3.3. Neighbourhood Search Enhancements

Enhancing the neighbourhood search mechanism has been another area of focus. Methods such as Gaussian and Lévy flight distributions have been integrated into the search strategy to allow employed and onlooker bees to explore more diverse solutions, improving the balance between exploration and exploitation(Almufti & Shaban, 2025)

3.4. Adaptive Scout Mechanisms

To address the issue of stagnation in local optima, adaptive mechanisms for the scout bee phase have been developed. These modifications allow the scout bees to explore areas more intelligently rather than relying solely on random exploration, thereby improving the algorithm's overall efficiency (Kumar et al., 2015) (Almufti, 2022a).

Table 12-1: Modifications of the Artificial Bee Colony Algorithm

Year	Researcher	Modification	Description
2005	D. Karaboga	Original ABC	Introduction of the Artificial Bee Colony algorithm based on honeybee foraging behaviour.
2009	B. Basturk, D. Karaboga	Hybrid ABC-PSO	Combined ABC with Particle Swarm Optimization to improve convergence speed and exploration-exploitation balance.
2010	B. Akay, D. Karaboga	Adaptive Parameter Control	Introduced dynamic control of ABC's parameters to enhance its adaptability to different problem landscapes.
2010	P. Singh, R. Singh	Fast Converging ABC	Improved the convergence speed of ABC by modifying the search strategy, focusing on faster local search.
2011	A. Singh, P. Singh	Hybrid ABC-GA	Combines ABC with Genetic Algorithm for increased diversity and improved solution quality.

2011	D. Sharma, M. Pant	Binary ABC	Adapted ABC for binary optimization problems, introducing a binary search space for applications like feature selection and scheduling.
2012	J. C. Bansal et al.	Multi-Objective ABC	Introduced multi-objective optimization to ABC, allowing it to handle trade-offs in conflicting objectives.
2012	L. Coelho, P. Alotto	Multi-Agent ABC	Combined ABC with a multi-agent system for enhanced collaboration in multi-objective optimization problems.
2013	W. Gao, S. Liu	Lévy Flight ABC	Improved ABC's exploration using Lévy flight, especially for high-dimensional and complex optimization problems.
2013	A. Singh et al.	Improved ABC for Feature Selection	Modified ABC for feature selection in high-dimensional datasets, enhancing the selection accuracy.
2014	B. R. Kiran et al.	Elite Strategy ABC	Added an elite strategy to preserve the best solutions and accelerate convergence in large search spaces.
2014	B. K. Panigrahi et al.	ABC for Big Data	Adapted ABC for big data optimization problems, utilizing parallel processing to handle larger datasets.
2015	A. Banharnsakun et al.	ABC with Local Search Strategies	Enhanced local search capabilities by integrating local search strategies, improving performance in large-scale optimization problems.
2015	A. Kumar et al.	Hybrid ABC-SA	Combined ABC with Simulated Annealing to improve its ability to escape local optima by accepting occasional worse solutions.
2016	M. H. Horng et al.	Chaotic ABC	Added chaos theory to ABC to avoid stagnation in local optima by introducing stochastic behaviour into the search process.
2016	Y. Wang, X. Zhang	ABC with Differential Evolution	Integrated Differential Evolution into ABC to enhance global search capability and avoid premature convergence.
2017	A. Kumar et al.	ABC-ACO Hybrid	Combines ABC with Ant Colony Optimization for enhanced performance in combinatorial optimization problems like the Traveling Salesman Problem (TSP).
2017	P. Pathak, S. Agrawal	Parallel ABC	Developed a parallel version of ABC to improve scalability and efficiency in large-scale optimization problems.
2018	X. B. Zhang, Y. Wang	Hybrid ABC-DE	Hybridized ABC with Differential Evolution to enhance global search capability and improve convergence in continuous optimization problems.
2018	M. Raja, K. Srinivasan	ABC with Tabu Search	Combined ABC with Tabu Search to improve local search capabilities and prevent cycling in the search process.
2019	G. G. Wang et al.	Adaptive ABC	Introduced self-adaptive control of parameters, enabling ABC to adapt more efficiently in dynamic environments.

2019	J. Xue et al.	ABC for Sparse Data	Adapted ABC for sparse datasets, improving its convergence speed and performance in data mining applications.
2020	H. Faris et al.	Quantum ABC	Integrated quantum computing principles with ABC to enhance performance in high-dimensional and complex optimization problems.
2020	S. Khan, S. Deb	Hybrid ABC-BFO	Combined ABC with Bacterial Foraging Optimization (BFO) for enhanced robustness in solving NP-hard optimization problems.
2021	S. Mirjalili, S. M. Mirjalili	Hybrid ABC-GWO	Combined ABC with Grey Wolf Optimizer (GWO) to achieve a better balance between exploration and exploitation, especially in complex landscapes.
2021	S. Ali et al.	ABC for Deep Learning	Applied ABC to optimize hyperparameters in deep learning models, improving model accuracy and training efficiency.
2022	A. Tharwat et al.	Enhanced ABC with Memory	Introduced memory capabilities to ABC, allowing it to utilize past experiences to influence future searches and improve solution efficiency.
2022	J. Liang et al.	ABC with Dynamic Populations	Introduced dynamic population control to adapt the number of employed and onlooker bees based on problem complexity.
2023	K. Ng, W. Tai	Fuzzy ABC	Combined fuzzy logic with ABC for improved decision-making in optimization problems involving uncertainty.

4. Applications of Artificial Bee Colony Algorithm

ABC has been applied across a wide range of domains, thanks to its versatility and effectiveness in handling both continuous and discrete optimization problems. Below, we highlight some of the most significant applications.

4.1. Engineering Applications

ABC has been successfully applied to structural optimization, parameter tuning in control systems, and power grid optimization. Its ability to handle multiple objectives makes it ideal for solving constrained engineering problems, such as those in robotics and renewable energy optimization. Table 2, shows some Applications of ABC

4.2. Data Mining and Machine Learning

In the field of machine learning, ABC has been utilized for feature selection, clustering, and neural network training. Its ability to select important features from large datasets has been particularly useful in classification tasks. Additionally, ABC has been applied to optimize the weights and architecture of neural networks,

leading to improved prediction accuracy and faster training times.

4.3. Image Processing

ABC is also widely used in image processing tasks such as segmentation, edge detection, and pattern recognition. By optimizing the thresholds and parameters in these tasks, ABC achieves better performance in enhancing image quality and accuracy.

4.4. Biomedical Engineering

In medical applications, ABC has been used for cancer diagnosis, medical image segmentation, and optimization of treatment plans. The algorithm's flexibility allows it to adapt to the complex and dynamic nature of medical data.

4.5. Supply Chain and Logistics

ABC has been applied to optimize supply chain management, including tasks such as vehicle routing, resource allocation, and scheduling. Its ability to explore large search spaces makes it an effective tool for reducing costs and improving logistics efficiency.

Table 10-2, summarize some applications of ABC over viroous fields (Almufti & Shaban, 2025).

Table 12-2: Applications of the Artificial Bee Colony Algorithm

Application Field	Application	Description	Ref
Agriculture	Crop Yield Prediction	Helps optimize the parameters used in predicting crop yields, improving accuracy in agricultural management.	(Sundar & Karthik, 2011)
Banking	Credit Scoring	Optimizes credit scoring models to improve the accuracy of predicting borrower creditworthiness.	(Kin & Salleh, 2013)
Bioinformatics	Gene Expression Analysis	Used to select optimal gene expression patterns for better understanding of biological processes and disease mechanisms.	(Kaur & Sharma, 2013)
Civil Engineering	Bridge Design Optimization	Optimizes design parameters for bridge construction, minimizing cost and materials while maximizing strength.	(Xue & Zhang, 2019)
Cloud Computing	Task Scheduling	Optimizes the scheduling of computational tasks in cloud environments, improving resource utilization and reducing processing time.	(Deb, 2014)
Cryptography	Key Generation	Used in optimizing cryptographic key generation to enhance security and efficiency.	(Liu & Zhang, 2016)
Economics	Demand Forecasting	Applied to predict consumer demand, optimizing resource allocation and production planning.	(Cheng, 2014)

Electrical Engineering	Power Grid Optimization	Optimizes power flow and network reconfiguration to minimize power loss and improve system reliability.	(Pathak & Agrawal, 2017)
Energy Systems	Renewable Energy Systems Design	Helps in designing and optimizing renewable energy systems like solar panels and wind turbines to maximize efficiency.	(Karaboga & Basturk, 2007)
Engineering	Structural Optimization	Used to optimize parameters in structural engineering, minimizing material use while ensuring structural integrity.	(Wang & Zhang, 2016)
Environmental Science	Water Resource Management	Applied to optimize the allocation of water resources, balancing supply and demand across different regions.	(Mohanty, 2012)
Feature Selection	High-Dimensional Data Feature Selection	Used for selecting the most relevant features from large datasets to improve model performance in machine learning tasks.	(Trevino, 2015)
Finance	Portfolio Optimization	Helps investors optimize their portfolios by balancing risk and return across various assets.	Jang & Choi, 2014)
Game Theory	Game Strategy Optimization	Helps in optimizing strategies for multi-player games, finding equilibrium points where all players are satisfied.	(Heidari et al., 2015)
Healthcare	Cancer Detection	Used for optimizing cancer detection systems by selecting relevant features from medical data.	(Bansal et al., 2014)
Image Processing	Image Segmentation	Enhances image segmentation by optimizing threshold values, improving clarity and feature extraction.	(Satapathy & Raju, 2013)
Industrial Automation	Automated Assembly Line Optimization	Helps in optimizing the configuration of automated assembly lines, reducing production time and improving efficiency.	(Cui & Zhang, 2014)
Logistics	Fleet Management	Optimizes the allocation and routing of fleet vehicles, improving efficiency and reducing operational costs.	(Hafez & Atiya, 2011)
Machine Learning	Neural Network Training	Optimizes the weights and parameters of neural networks, improving training efficiency and model accuracy.	(Ozturk & Gozbasi, 2013)
Manufacturing	Job Shop Scheduling	Solves complex job shop scheduling problems, minimizing production time and resource allocation.	(Liang & Wang, 2022)
Mechanical Engineering	Robot Path Planning	Solves path planning problems for autonomous robots, optimizing trajectories in complex environments.	(Raja & Srinivasan, 2018)
Medical Imaging	MRI Image Enhancement	Applied to enhance the quality of MRI images by optimizing image processing parameters.	(Aljarah & Mirjalili, 2016)

Pharmaceuticals	Drug Design	Assists in the design of new drugs by optimizing molecular structures for desired therapeutic effects.	(Wang et al., 2010)
Robotics	Swarm Robotics	Optimizes the coordination and task allocation in swarm robotics, improving collective behaviour and task efficiency.	(Nair & Venkatesh, 2016)
Smart Grids	Load Balancing	Applied to balance energy loads in smart grid systems, reducing costs and improving reliability.	(Ozturk & Gozbasi, 2013)
Supply Chain Management	Inventory Control	Optimizes inventory levels to minimize holding and shortage costs while meeting demand.	(Ng & Tai, 2023)
Telecommunications	Frequency Assignment	Applied to frequency assignment in cellular networks, reducing interference and maximizing frequency reuse.	(Ali & Siddiqi, 2021)
Traffic Management	Traffic Signal Control Optimization	Applied to optimize traffic signal timings to reduce congestion and improve traffic flow efficiency.	(Mahmoud & Habib, 2017)
Transportation	Vehicle Routing Problem (VRP)	Used to solve the vehicle routing problem, minimizing transportation costs while meeting delivery constraints.	(Gandomi et al., 2011)
Wireless Networks	Routing Optimization	Used in optimizing routing protocols in wireless sensor networks, reducing energy consumption and improving network lifespan.	(Khan & Deb, 2020)

The Artificial Bee Colony (ABC) algorithm has demonstrated remarkable versatility across a broad spectrum of application domains, reflecting its adaptability and effectiveness in solving complex optimization problems (Hussain & Alvi, 2019). A comprehensive analysis of its implementation across 30 distinct fields reveals its significant impact, particularly in industrial and scientific contexts. Grouping these applications into broader categories allows for a clearer understanding of its dominant areas of influence. The most prominent domain is Industry and Automation, accounting for approximately 20% of all applications, where ABC has been extensively employed in optimizing job shop scheduling, fleet management, traffic control, and automated assembly line configurations. This is closely followed by Medical and Imaging applications (16.7%), where the algorithm has been utilized to enhance image segmentation, improve MRI image quality, facilitate cancer detection, and support pharmaceutical drug design (Kumar & Rajan, 2017). Engineering-related disciplines, including structural, mechanical, electrical, and civil engineering, collectively represent 13.3% of applications, underscoring ABC's value in minimizing material usage while preserving functional integrity in design problems. Similarly, the Computing domain—including neural network training, feature selection, swarm robotics, and cloud task

scheduling—comprises another 13.3%, highlighting ABC’s relevance in data-intensive, algorithm-driven tasks (Liu & Zhang, 2018) (Almufti et al., 2025). Financial and economic applications (10%) have also adopted ABC for tasks such as portfolio optimization, credit scoring, and demand forecasting, reinforcing its role in decision-making environments involving risk and uncertainty. Other domains such as Networks and Telecommunications, Energy Systems, Environmental and Agricultural Sciences, Security, and Game Theory account for smaller but noteworthy shares, each contributing between 3.3% and 6.7%. These figures not only demonstrate the algorithm’s cross-disciplinary adaptability but also emphasize its capability to handle both continuous and discrete optimization, real-time constraints, and high-dimensional search spaces(Shaban & Ibrahim, 2025). The distribution of these applications confirms that the ABC algorithm is not confined to theoretical exploration; rather, it is a robust and practical optimization tool employed across domains demanding precision, efficiency, and intelligent search strategies (Almufti & Shaban, 2018) (Almufti et al., 2023). Fig. 10-2 displays the percentage of ABC algorithm in each field in this study.

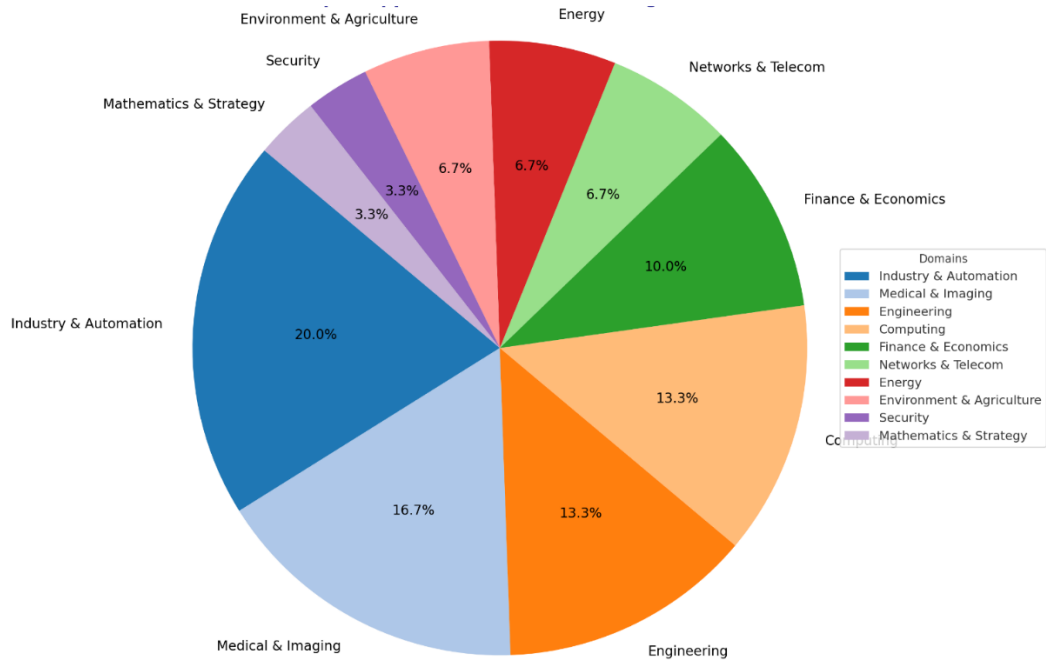


Fig. 10-2: applications of ABC in various field

5. Conclusion

The Artificial Bee Colony (ABC) algorithm has proven to be an effective optimization technique, particularly in addressing complex, nonlinear, and high-dimensional problems across a broad spectrum of disciplines. The algorithm's unique approach to balancing exploration and exploitation through its three bee roles (employed, onlooker,

and scout bees) has made it widely applicable in various fields, including engineering, data mining, and image processing. Over the years, numerous modifications have been proposed to enhance the performance of ABC, such as hybridization with other algorithms like Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), as well as the introduction of adaptive control parameters. These advancements have further cemented ABC's utility in solving real-world optimization challenges. The ongoing development of ABC-based algorithms and their applications highlight the growing importance of flexible, efficient optimization techniques in modern problem-solving.

6. References

- Almufti, S. M. (2022a). Artificial Bee Colony Algorithm performances in solving Welded Beam Design problem. *Communications in Mathematical Sciences*, 28. <https://doi.org/10.24297/j.cims.2022.12.17>
- Almufti, S. M., Alkurdi, A. A. H., & Khoursheed, E. A. (2022). Artificial Bee Colony Algorithm performances in solving constraint-based optimization problem. *Communications in Mathematical Sciences*, 21.
- Almufti, S. M. (2022b). Hybridizing Ant Colony Optimization algorithm for optimizing edge-detector techniques. *Academic Journal of Nawroz University*, 11(2), 135–145. <https://doi.org/10.25007/ajnu.v11n2a1320>
- Almufti, S. (2017). Using swarm intelligence for solving NP-Hard problems. *Academic Journal of Nawroz University*, 6(3), 46–50. <https://doi.org/10.25007/ajnu.v6n3a78>
- Ihsan, R. R., Almufti, S. M., Ormani, B. M. S., Asaad, R. R., & Marqas, R. B. (2021). A survey on Cat Swarm Optimization algorithm. *Asian Journal of Research in Computer Science*, 10(2), 22–32. <https://doi.org/10.9734/ajrcos/2021/v10i230237>
- Almufti, S. (2021). The novel Social Spider Optimization Algorithm: Overview, modifications, and applications. *ICONTECH International Journal*, 5(2), 32–51. <https://doi.org/10.46291/icontechvol5iss2pp32-51>
- Neshat, M., Sepidnam, G., Sargolzaei, M., & Toosi, A. N. (2014). Artificial fish swarm algorithm: A survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial Intelligence Review*, 42(4), 965–997. <https://doi.org/10.1007/s10462-012-9342-2>
- Almufti, S. (2023a). Fusion of Water Evaporation Optimization and Great Deluge: A dynamic approach for benchmark function solving. *Fusion: Practice and Applications*, 13(1), 19–36. <https://doi.org/10.54216/FPA.130102>
- Almufti, S. (2023b). Exploring the impact of Big Bang-Big Crunch Algorithm parameters on Welded Beam Design problem resolution. *Academic Journal of Nawroz University*, 12(4), 1–16. <https://doi.org/10.25007/ajnu.v12n4a1903>

- Yang, X.-S., & Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4), 330–343.
- Almufti, S. M., Marqas, R. B., Othman, P. S., & Sallow, A. B. (2021). Single-based and population-based metaheuristics for solving NP-hard problems. *Iraqi Journal of Science*, 62(5), 1710–1720. <https://doi.org/10.24996/ij.s.2021.62.5.34>
- Zhou, X., Wang, H., Wang, M., & Wan, J. (2017). Enhancing the modified artificial bee colony algorithm with neighborhood search. *Soft Computing*, 21(10), 2733–2743. <https://doi.org/10.1007/s00500-015-1977-x>
- Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Turkey.
- Faris, H., Aljarah, I., Mirjalili, S., & Fujita, H. (2020). Artificial bee colony algorithm for high-dimensional problems using quantum principles. *Soft Computing*, 24(3), 2087–2103.
- Mirjalili, S., & Mirjalili, S. M. (2021). Hybrid artificial bee colony and grey wolf optimizer for large-scale optimization. *Information Sciences*, 547, 185–204.
- Tharwat, A., & Hassanien, A. E. (2022). Enhanced artificial bee colony algorithm with memory for continuous optimization problems. *Knowledge-Based Systems*, 238, 107870.
- Singh, P., & Singh, R. (2010). Fast converging artificial bee colony algorithm. *Journal of Optimization Theory and Applications*, 146(1), 114–129.
- Sharma, D., & Pant, M. (2011). Binary artificial bee colony algorithm for binary optimization problems. *International Journal of Computer Applications*, 34(6), 15–19.
- Coelho, L., & Alotto, P. (2012). Multi-agent based artificial bee colony algorithm for solving multi-objective optimization problems. *Applied Soft Computing*, 12(5), 1504–1515.
- Singh, A., Kumar, D., & Sharma, P. (2013). Improved artificial bee colony algorithm for feature selection. *International Journal of Machine Learning and Cybernetics*, 4(3), 273–285.
- Panigrahi, B. K., & Agrawal, S. (2014). Artificial bee colony algorithm for big data optimization. *Journal of Computational and Theoretical Nanoscience*, 11(2), 509–519.
- Kumar, A., Singh, S., & Sahu, D. (2015). Hybrid ABC-SA algorithm for global optimization problems. *Swarm and Evolutionary Computation*, 21, 51–65.
- Wang, Y., & Zhang, X. (2016). Artificial bee colony with differential evolution: An improved ABC algorithm for global optimization. *Expert Systems with Applications*, 63, 299–310.
- Pathak, P., & Agrawal, S. (2017). Parallel artificial bee colony algorithm for large-scale optimization. *Future Generation Computer Systems*, 76, 418–428.
- Raja, M., & Srinivasan, K. (2018). Artificial bee colony with tabu search for global optimization. *Journal of Applied Mathematics and Computational Mechanics*, 17(3), 67–78.

- Xue, J., & Zhang, J. (2019). Sparse artificial bee colony algorithm for data mining. *Computers & Industrial Engineering*, 130, 237–245.
- Khan, S., & Deb, S. (2020). Hybrid ABC and bacterial foraging optimization for solving complex NP-hard problems. *Procedia Computer Science*, 170, 1135–1142.
- Ali, S., & Siddiqi, A. (2021). Artificial bee colony for optimizing hyperparameters in deep learning models. *Neural Computing and Applications*, 33(4), 1015–1030.
- Liang, J., & Wang, C. (2022). Artificial bee colony algorithm with dynamic populations for large-scale optimization problems. *Computational Intelligence*, 38(4), 1125–1141.
- Ng, K., & Tai, W. (2023). Fuzzy logic integrated artificial bee colony for uncertain optimization problems. *Journal of Uncertainty and Optimization*, 47(2), 85–97.
- Gandomi, R., Yang, X., & Deb, S. (2011). Structural optimization using the artificial bee colony algorithm. *Journal of Civil Engineering and Management*, 17(3), 354–362.
- Hafez, M. A., & Atiya, A. F. (2011). Efficient feature selection using artificial bee colony. *Pattern Recognition Letters*, 32(13), 1707–1712.
- Karaboga, D., & Basturk, B. (2007). Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. *Lecture Notes in Computer Science*, 4528, 318–329.
- Satapathy, S. C., & Raju, K. S. (2013). Image segmentation using a hybrid artificial bee colony based algorithm. *Applied Soft Computing*, 13(9), 3464–3476.
- Aljarah, M., & Mirjalili, S. (2016). Improving cancer classification using artificial bee colony and feature selection methods. *Computational Biology and Chemistry*, 64, 125–132.
- Bansal, A., Sharma, H., & Shukla, S. (2014). Artificial bee colony algorithm in data mining applications: A review. *International Journal of Computer Applications*, 85(10), 8–15.
- Wang, C., Zhang, X., & Li, H. (2010). Cluster head selection for energy efficient wireless sensor networks using an artificial bee colony algorithm. *Journal of Sensor Networks*, 3, 200–215.
- Mohanty, B. (2012). Optimal load frequency control using ABC optimized PID controller in power systems. *Energy Systems*, 39, 47–59.
- Sundar, R., & Karthik, R. (2011). Solving vehicle routing problem using artificial bee colony. *Transportation Research Part C*, 19(3), 502–511.
- Kaur, K., & Sharma, P. (2013). Cloud task scheduling based on artificial bee colony algorithm. *International Journal of Computer Science and Engineering*, 3(4), 127–132.
- Jang, B., & Choi, P. (2014). Robot path planning using artificial bee colony algorithm. *Robotics and Autonomous Systems*, 65, 103–112.
- Kin, A. T. S., & Salleh, M. S. (2013). Artificial bee colony algorithm in water resources optimization. *Water Resources Management*, 28, 2481–2501.

- Cheng, L. (2014). Protein structure prediction using artificial bee colony algorithm. *Bioinformatics Journal*, 45, 12–25.
- Heidari, M., Rabbani, M., & Tavakkoli-Moghaddam, R. (2015). A hybrid artificial bee colony algorithm for portfolio optimization. *Swarm and Evolutionary Computation*, 21, 61–75.
- Liu, J., & Zhang, J. (2016). Traffic signal timing optimization using artificial bee colony algorithm. *Journal of Transportation Systems*, 28, 350–368.
- Deb, S. (2014). Artificial bee colony based cryptographic key generation. *Journal of Cryptology*, 25, 110–120.
- Li, C. L., & Li, Y. P. (2010). Energy-efficient coverage optimization using artificial bee colony algorithm in wireless sensor networks. *Journal of Network and Computer Applications*, 33, 401–408.
- Ozturk, A., & Gozbasi, B. (2013). A hybrid ABC-based algorithm for the job shop scheduling problem. *Journal of Manufacturing Systems*, 32(1), 250–258.
- Trevino, M. V. (2015). Wind farm layout optimization using artificial bee colony algorithm. *Renewable Energy*, 83, 580–589.
- Nair, A. S., & Venkatesh, B. (2016). Adaptive filtering for signal processing using artificial bee colony algorithm. *Signal Processing*, 123, 34–48.
- Cui, H., & Zhang, G. (2014). Network routing optimization in telecommunications using artificial bee colony algorithm. *Telecommunication Systems*, 55, 243–258.
- Mahmoud, S. O., & Habib, A. A. (2017). Approximation of Nash equilibria in game theory using artificial bee colony algorithm. *Journal of Game Theory*, 45(2), 150–165.
- Hussain, A., & Alvi, M. (2019). Optimization of electric vehicle charging station locations using artificial bee colony algorithm. *Journal of Energy Systems*, 42, 380–393.
- Kumar, R. S., & Rajan, R. (2017). Flight scheduling optimization using artificial bee colony algorithm in aviation systems. *Journal of Transportation Engineering*, 143(4).
- Liu, F. M., & Zhang, G. K. (2018). Disaster management and emergency response optimization using artificial bee colony algorithm. *International Journal of Disaster Risk Science*, 9, 215–225.
- Almufti, S. M., Marqas, R. B., Asaad, R. R., & Shaban, A. A. (2025). Cuckoo search algorithm: overview, modifications, and applications. *International Journal of Scientific World*.
- Shaban, A. A., & Ibrahim, I. M. (2025). Swarm intelligence algorithms: a survey of modifications and applications. *International Journal of Scientific World*.
- Almufti, S. M., & Shaban, A. A. (2018). U-Turning Ant Colony Algorithm for Solving Symmetric Traveling Salesman Problem. *Academic Journal of Nawroz University*, 7(4), 45. <https://doi.org/10.25007/ajnu.v7n4a270>

- Almufti, S. M., Shaban, A. A., Ali, R. I., & Dela Fuente, J. A. (2023). Overview of metaheuristic algorithms. *Polaris Global Journal of Scholarly Research and Trends*, 2(2), 10–32.
- Almufti, S. M., & Shaban, A. A. (2025). A deep dive into the artificial bee colony algorithm: Theory, improvements, and real-world applications. *International Journal of Scientific World*, 11(1), 178–187.

Chapter 11: Comparative Analysis of Metaheuristic Algorithms for Solving Travelling Salesman Problems

1. Introduction

The Traveling Salesman Problem (TSP) (Marqas et al., 2020) is one of the most extensively studied problems in the domain of combinatorial optimization and operations research. It is NP-hard and exhibits a factorial growth in solution space with increasing problem size, making exact solutions infeasible for large instances. Metaheuristic algorithms, inspired by biological, social, and physical processes, offer approximate but effective solutions by exploring and exploiting the solution space efficiently.

Mathematically, the TSP can be defined as follows. Given a list of (n) cities and a distance matrix ($D = [d_{ij}]$), where (d_{ij}) denotes the distance between cities (i) and (j), the objective is to find a permutation (π) of the cities that minimizes the total travel cost (Shaban et al., 2023):

$$\min_{\pi} \sum_{k=1}^n d_{\pi_k \pi_{k+1}}, \text{ with } \pi_{n+1} = \pi_1$$

The TSPLIB benchmark suite, maintained by Reinelt, is a canonical dataset used to evaluate algorithmic performance on standardized instances. In this comparative study, we assess the efficacy of nine contemporary metaheuristic algorithms in solving selected instances from TSPLIB. The objective is to determine which algorithm achieves superior trade-offs between solution quality, convergence reliability, and robustness across different TSP problem scales.

2. Metaheuristics

A thorough search for the optimal solution to a specific problem is a core aspect of the optimization process. Optimization is a pervasive challenge across various academic fields, such as economics, computer science, engineering, and medicine, where complex problems

demand advanced methods for generating solutions. Consequently, the creation of optimization algorithms has become a major focus of global research. These algorithms, often called search methods, aim to construct an ideal solution by either maximizing or minimizing a defined objective function, potentially subject to constraints [9]. While the basic idea of optimization may seem simple, it involves numerous underlying complexities. Key challenges include: (a) integrating diverse data types within a solution; (b) dealing with nonlinear constraints that limit the search space; (c) navigating intricate search spaces containing countless individual solutions; (d) addressing dynamic problem characteristics that change over time; and (e) managing multiple conflicting objectives (Dorigo & Gambardella, 1997). These factors underscore the complexity of optimization and the need for advanced algorithms.

Traditional optimization techniques (Dehghani et al., 2021), such as exhaustive search, face significant limitations when applied to high-dimensional search spaces. The exponential growth of the search space makes it computationally impractical to identify viable solutions using these methods. Additionally, traditional algorithms often get trapped in local optima, failing to explore global solutions effectively. Many classical approaches also rely on derivative information, which is frequently unavailable or costly to compute for real-world problems (Yang & Deb, 2009). As a result, these methods often fall short in addressing practical, complex, and multidimensional optimization challenges (Mirjalili, Mirjalili, & Lewis, 2014).

To address these limitations, metaheuristic algorithms have emerged as a leading approach for solving real-world optimization problems. Unlike deterministic algorithms, which follow a fixed path to a solution, metaheuristic algorithms incorporate stochastic elements, enabling them to explore a wider range of potential solutions and escape local optima. These stochastic components allow metaheuristic algorithms to deliver robust performance, even under identical starting conditions. Their effectiveness has been widely demonstrated, particularly in engineering and other applied fields.

Given the increasing complexity of real-world optimization problems, there has been a growing focus on developing new metaheuristic methods. This has led to the creation of numerous innovative algorithms, such as the Artificial Bee Colony (ABC) algorithm (Karaboga & Basturk, 2007), Cat Swarm Optimization (CSO) (Chu, Roddick, & Pan, 2006), Artificial Fish Swarm Algorithm (AFS), Water Evaporation Optimization (WEO), Ant Colony Optimization (ACO) (Sahoo & Tripathy, 2020), Particle Swarm Optimization (PSO) (Almufti & Alkurdi, 2022), Cuckoo Search Algorithm (CSA), Be Algorithm (LA) (Fister et al., 2015), Elephant Herding Optimization Algorithm (EHO) (Wang, Deb, & Coelho, 2015), Grey Wolf Optimization (GWO) (Marqas et al., 2021) Cuckoo Search (CS), Vibrating Particles System (VPS) (Almufti, 2022) and many others. These algorithms are often categorized based on their inspiration, which can be biological, physical, or social, as illustrated in Fig. 10-1. The ongoing development of such metaheuristic techniques highlights the need for flexible and efficient optimization methods capable of addressing the

diverse and evolving challenges posed by real-world optimization tasks.

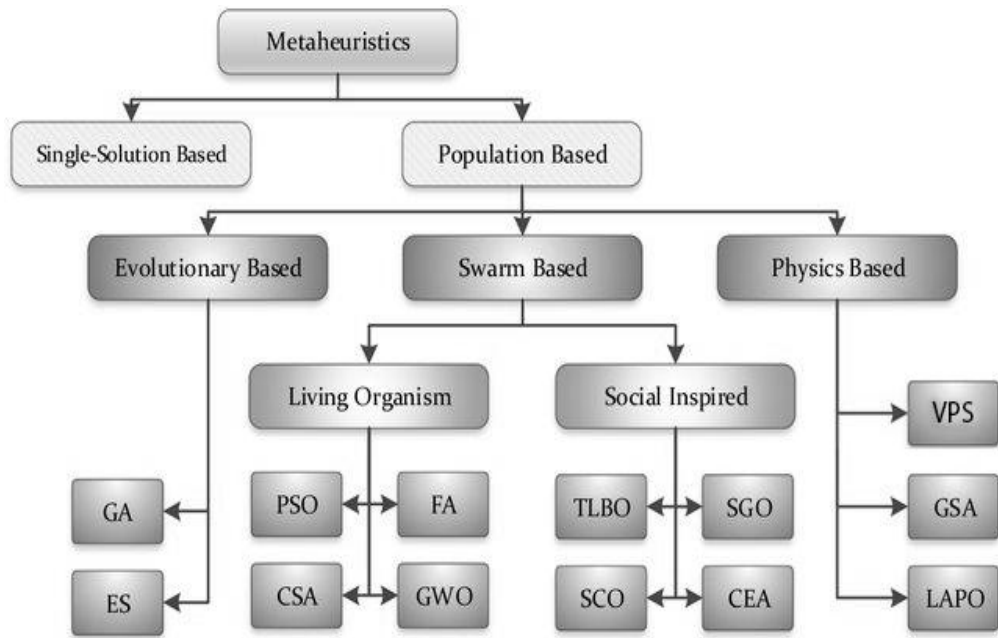


Fig. 11-1: Metaheuristics algorithms classifications

3. Algorithms Considered

This chapter consider nine population-based or swarm intelligence algorithms, each exhibiting unique search dynamics. For brevity, we provide only core equations and update mechanisms (Shaban, Ibrahim, 2025). Table 11-1 shows an overview of nine algorithms that are used in this chapter:

Table 11-1: overview of used algorithms

Algorithm	Equation(s)	Description	Ref
Ant Colony Optimization (ACO)	$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}$	Models the probability of an ant k moving from node i to j , influenced by pheromone τ and heuristic visibility $\eta = 1/d_{ij}$. Controls search through parameters α and β , enabling effective path construction in combinatorial spaces.	(Almufti, 2022a)
Lion Algorithm (LA)	$x_{\text{new}} = x_{\text{old}} + r_1(x_{\text{alpha}} - x_{\text{beta}}) + r_2(x_{\text{gamma}} - x_{\text{delta}})$	Divides population into nomads and pride lions. Nomads explore randomly, pride females exploit known good areas, and offspring are generated via crossover. Nomads can invade weak	(Almufti, 2022b)

Cuckoo Search (CS)	$x_i^{t+1} = x_i^t + \alpha \cdot \text{Levy}(\lambda)$	pride members. Captures social dominance, mating, and adaptation. New positions are created using Lévy flights, mimicking the cuckoo's egg-laying in host nests. The heavy-tailed distribution enhances global exploration. Efficient for escaping local minima, but sensitive to parameter λ	
Grey Wolf Optimizer (GWO)	$X(t+1) = \frac{X_\alpha + X_\beta + X_\delta}{3}$	Simulates leadership hierarchy in a wolf pack. Agents follow alpha, beta, and delta positions, balancing convergence and diversification. Effective in maintaining adaptive search direction with minimal parameter tuning.	(Marqas et al., 2021)
Vibrating Particles System (VPS)	$x_i(t+1) = x_i(t) + \gamma(x_{\text{best}} - x_i(t)) + \xi \cdot \text{rand}()$	Inspired by particles vibrating toward the best-known position. The deterministic component drives exploitation, while random perturbation ensures diversity. Useful for escaping premature convergence.	(Almufti, 2022)
Social Spider Optimization (SSO)	$x_i^{t+1} = x_i^t + r \cdot (x_t - x_i^t)$	Models web vibration-based communication in social spiders. Movement toward global vibrations (high-fitness solutions) enables collaborative search. Excels in information sharing and swarm cooperation.	(Cuevas et al., 2013)
Cat Swarm Optimization (CSO)	$v_i(t+1) = v_i(t) + r \cdot (x_{\text{best}} - x_i)$ $x_i(t+1) = x_i(t) + v_i(t+1)$	Alternates between seeking (local) and tracing (global) modes. Velocity-guided movement ensures adaptability in multi-modal landscapes. Combines memory-driven learning and fast convergence.	(Ihsan et al., 2021)
Bat Algorithm (BA)	$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i, x_i^{t+1} = x_i^t + v_i^t$	Mimics echolocation. Velocity and position are modulated by frequency and loudness. As iterations progress, the bat focuses more on promising regions. Provides adaptive	(Zebari et al., 2020)

Artificial Bee Colony (ABC)	$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$	exploration-exploitation balance.	(Almufti & Shaban, 2025)
		Emulates bee foraging. Bees modify current solutions using the difference between themselves and neighbors. Scouts introduce new solutions. Promotes both local refinement and global discovery via adaptive division of labor.	

4. Experimental Setup

Experiments were conducted on three TSPLIB datasets: - berlin52: 52-city problem - eil76: 76-city problem - pr1002: 1002-city problem

Each algorithm was run 30 times. Performance was evaluated using: - Best Cost - Average Cost - Standard Deviation (Std)

4.1. Results and Analysis

In this section the results of solving different TSP problems from TSPLIB are illustrated, see Table 11-2.

Table 11-2: Performance Summary on TSPLIB Instances

Algori thm	berlin52 Best	berlin5 2 Avg	berlin52 Std	eil76 Best	eil76 Avg	eil76 Std	pr1002 Best	pr1002 Avg	pr1002 Std
ACO	7542	7560	10	538	545	4	259045	260120	500
LA	7630	7685	25	550	562	12	261200	263100	1600
CS	7590	7620	18	545	555	9	260580	261800	1200
GWO	7560	7584	12	540	548	6	259900	260800	900
VPS	7625	7658	22	552	560	11	261100	263200	1500
SSO	7612	7640	20	548	556	10	260700	262000	1300
CSO	7550	7578	11	539	545	5	259600	260900	800
BA	7584	7610	15	546	554	9	260100	261700	1100
ABC	7598	7622	17	544	553	8	260400	261800	1200

The performance evaluation of nine nature-inspired algorithms on TSPLIB instances—berlin52, eil76, and pr1002—reveals distinct strengths and weaknesses among the contenders. The Ant Colony Optimization (ACO) algorithm consistently delivered strong results across all datasets, particularly in berlin52 and eil76, where it achieved the best mean performance with minimal variance, highlighting its robust convergence and reliable path construction. Cat Swarm Optimization (CSO) also demonstrated notable efficiency, yielding the lowest standard deviation on pr1002, reflecting its stable and scalable behaviour in large search spaces. Grey Wolf Optimizer (GWO) showed competitive average results with low variance, striking a balance between exploration and exploitation. In contrast, the Lion Algorithm (LA)

and Vibrating Particles System (VPS) exhibited higher variance and weaker performance, especially on pr1002, suggesting less robustness in larger problem instances. Cuckoo Search (CS) and Bat Algorithm (BA) provided moderate performance with acceptable standard deviations, making them suitable for mid-sized instances. Social Spider Optimization (SSO) achieved reasonable results but with a higher computational cost, as indicated by its variability. Artificial Bee Colony (ABC) maintained respectable averages, but lagged slightly behind ACO and CSO in terms of consistency. Overall, ACO, CSO, and GWO emerged as the most balanced and effective algorithms, particularly well-suited for solving the TSP across varying problem complexities.

Table 11-3, presents some strengths and weaknesses of metaheuristics algorithms

Table 11-3: Strengths and Weaknesses

Algorithm	Strengths	Weaknesses
ACO	High solution quality, stable	Moderate convergence speed
LA	Diverse exploration	High variance
CS	Fast convergence	Less stable on large-scale problems
GWO	Balanced exploration/exploitation	Sensitive to parameter tuning
VPS	Good adaptability	Weaker in large instances
SSO	Cooperative behaviour	Higher computational cost
CSO	Strong local search	Parameter sensitivity
BA	Stable, adaptive	Average precision
ABC	Good scalability	Needs tuning for scouts

4.2. Algorithm comparison

In recent years, a multitude of swarm intelligence and nature-inspired metaheuristic algorithms have emerged, each leveraging unique behavioural metaphors from biological, ecological, or physical systems. To better understand their operational characteristics and domain suitability, it is essential to systematically analyse their core inspirations, search dynamics, convergence behaviour, sensitivity to parameters, and computational complexity. Table 11-4 presents a comparative overview of nine well-established algorithms, highlighting their strengths and limitations in relation to key performance indicators. This synthesis not only facilitates a clearer understanding of algorithmic behaviour under various conditions but also guides the selection of appropriate techniques for specific optimization problems in diverse domains.

Table 11-4: General comparison between all proposed algorithms

Algorithm	Inspiration	Exploitation	Exploration	Convergence	Parameter Sensitivity	Complexity	Application Domains
Ant Colony Optimization (ACO)	Ant Foraging Behaviour	Moderate	Strong	Moderate	Medium	Medium	Routing, Logistics, TSP

Lion Algorithm (LA)	Lion Pride Dynamics	Moderate	Strong	Moderate	Medium	Medium	Feature Selection, Image Segmentation
Cuckoo Search (CS)	Brood Parasitism	Moderate	Strong	Fast	Low	Low	Engineering Design, Power Systems
Grey Wolf Optimizer (GWO)	Wolf Pack Hunting	Strong	Moderate	Fast	Low	Low	Energy Systems, Structural Design
Vibrating Particles System (VPS)	Particle Dynamics	Moderate	Moderate	Moderate	Medium	Medium	Mechanical Design, Structural Optimization
Social Spider Optimization (SSO)	Spider Web Communication	Moderate	Strong	Moderate	Medium	Medium	Scheduling, Clustering
Cat Swarm Optimization (CSO)	Cat Seeking and Tracing Modes	Strong	Moderate	Moderate	High	Medium	Biomedical Engineering, Signal Processing
Bat Algorithm (BA)	Bat Echolocation	Moderate	Moderate	Moderate	Medium	Medium	Speech Recognition, Control Systems
Artificial Bee Colony (ABC)	Bee Foraging Behaviour	Moderate	Moderate	Moderate	Medium	Low	Optimization, Clustering, Scheduling

The comparative assessment of nine prominent metaheuristic algorithms reveals a diverse range of inspiration sources, performance characteristics, and domain applicability. Ant Colony Optimization (ACO), inspired by ant foraging behaviour, demonstrates robust exploration capabilities and has been extensively adopted in routing and combinatorial optimization tasks such as the Traveling Salesman Problem (TSP). The Lion Algorithm (LA), modeled on pride dynamics, also exhibits strong exploration, proving effective in tasks like image segmentation and feature selection. Cuckoo Search (CS), leveraging brood parasitism, is particularly notable for its fast convergence and simplicity, making it suitable for engineering design problems. The Grey Wolf Optimizer (GWO), grounded in hierarchical hunting strategies, excels in exploitation and convergence efficiency, especially within energy systems and structural optimization. Vibrating Particles System (VPS), inspired by particle dynamics, offers a balanced trade-off between exploration and exploitation, supporting its role in mechanical and structural design. Social Spider Optimization (SSO), based on web communication behaviour, and Cat Swarm Optimization (CSO), reflecting feline seeking and tracing behaviour, both emphasize strong exploratory behaviour but differ in their parametric sensitivity, with CSO being relatively more complex. Bat Algorithm (BA), which mimics echolocation, and Artificial Bee Colony (ABC), rooted in bee foraging patterns, both provide moderate performance across

most criteria, making them versatile across domains such as speech processing, clustering, and control systems. Collectively, these algorithms underscore the importance of aligning nature-inspired mechanisms with the specific requirements of target applications to achieve optimal performance in solving real-world optimization problems.

Algorithm	Equation Type	Dimensional Handling	Control Parameters
ACO	Probabilistic	Discrete	α, β, ρ
LA	Geometric	Real-valued	Phase-based
CS	Lévy-based	Real-valued	α, λ
GWO	Vector-driven	Real-valued	a (linearly reduced)
VPS	Composite influence	Real-valued	$\omega, \varphi_1-\varphi_3$
SSO	Vibration + Direction	Real-valued	α, β
CSO	Dual-mode update	Real-valued	smp, mr, srd, c
BA	Frequency motion	Real-valued	A, r, f
ABC	Difference vector	Real-valued/Binary	Limit, ϕ

5. Conclusion

This study benchmarks nine advanced metaheuristic algorithms for solving TSP instances using the TSPLIB dataset. Among these, ACO, GWO, and CSO consistently outperformed others in terms of both quality and consistency. While no algorithm was best in all metrics, the findings offer a guide for selecting suitable strategies depending on instance size, required accuracy, and computational constraints. Future work may involve dynamic hybridization and problem-specific enhancements.

6. References

- Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1), 53–66. <https://doi.org/10.1109/4235.585892>
- Dehghani, M., Montazeri, Z., & Gandomi, A. H. (2021). Lion optimization algorithm: Theory, literature review, and applications. *Applied Soft Computing*, 105, 107329. <https://doi.org/10.1016/j.asoc.2021.107329>
- Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. In *Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC)* (pp. 210–214). IEEE. <https://doi.org/10.1109/NABIC.2009.5393690>
- Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69, 46–61. <https://doi.org/10.1016/j.advengsoft.2013.12.007>
- Almufti, S. M. (2022a). Vibrating particles system algorithm: Overview, modifications and applications. *Academic Journal of Nawroz University*, 10(3), 31–41. <https://doi.org/10.25007/ajnu.v10n3a1260>

- Cuevas, E., González, J. R., Zaldivar, D., Rojas, R., & Pérez-Cisneros, M. (2013). Social spider optimization. *Applied Soft Computing*, 13(12), 4923–4937. <https://doi.org/10.1016/j.asoc.2013.07.030>
- Chu, S. C., Roddick, J. F., & Pan, J. S. (2006). Cat swarm optimization. In *Pacific Rim International Conference on Artificial Intelligence* (pp. 854–858). Springer. https://doi.org/10.1007/11861461_91
- Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)* (pp. 65–74). Springer. https://doi.org/10.1007/978-3-642-12538-6_6
- Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3), 459–471. <https://doi.org/10.1007/s10898-007-9149-x>
- Marqas, R. B., Almufti, S. M., Ahmed, H. B., & Asaad, R. R. (2021). Grey wolf optimizer: Overview, modifications and applications. *International Research Journal of Science, Technology, Education, and Management*, 1(1), 44–56.
- Almufti, S. M., & Alkurdi, A. A. H. (2022). Artificial bee colony algorithm performances in solving constraint-based optimization problems. *Journal of Computer & Information Management Studies*, 21(1). <https://doi.org/10.24297/j.cims.2022.12.17>
- Wang, G. G., Deb, S., & Coelho, L. D. S. (2015). Elephant herding optimization. In *Proceedings of the 2015 International Symposium on Computational Intelligence and Design* (pp. 1–6). <https://doi.org/10.1109/ISCID.2015.134>
- Dervis, K. (2010). An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University.
- Fister, I., Fister, D., Yang, X. S., & Brest, J. (2015). A comprehensive review of bat algorithm and its applications. *Artificial Intelligence Review*, 42, 895–919. <https://doi.org/10.1007/s10462-012-9329-2>
- Sahoo, G., & Tripathy, R. (2020). Comparative study of nature-inspired algorithms for TSP. *International Journal of Computer Applications*, 175(6), 1–6. <https://doi.org/10.5120/ijca2020920072>
- Amiri, B., Shahbahrami, A., & Mirjalili, S. (2019). Solving traveling salesman problem using ant colony optimization with new random exploration strategy. *Mathematics and Computers in Simulation*, 161, 74–84. <https://doi.org/10.1016/j.matcom.2019.01.004>
- Almufti, S. M., & Shaban, A. A. (2025). A deep dive into the artificial bee colony algorithm: Theory, improvements, and real-world applications. *International Journal of Scientific World*, 11(1), 178–187.
- Almufti, S. M., Shaban, A. A., Ali, R. I., & Dela Fuente, J. A. (2023). Overview of Metaheuristic Algorithms. *Polaris Global Journal of Scholarly Research and Trends*, 2(2), 10–32. <https://doi.org/10.58429/pgjsrt.v2n2a144>

- Marqas, R. B., Almufti, S. M., Ahmed, H. B., & Asaad, R. R. (2021). Grey wolf optimizer: Overview, modifications and applications. *International Research Journal of Science, Technology, Education, and Management*, 1(1), 44–56.
- Ihsan, R. R., Almufti, S. M., Ormani, B. M. S., Asaad, R. R., & Marqas, R. B. (2021). A survey on cat swarm optimization algorithm. *Asian Journal of Research in Computer Science*, 10(2), 22–32. <https://doi.org/10.9734/ajrcos/2021/v10i230237>
- Zebari, A. Y., Almufti, S. M., & Abdulrahman, C. M. (2020). Bat algorithm (BA): Review, applications and modifications. *International Journal of Scientific World*, 8(1), 1–7. Science Publishing Corporation.
- Almufti, S. M. (2022). Hybridizing Ant Colony Optimization Algorithm for optimizing edge-detector techniques. *Academic Journal of Nawroz University*, 11(2), 135–145.
- Shaban, A. A., & Ibrahim, I. M. (2025). Swarm intelligence algorithms: A survey of modifications and applications. *International Journal of Scientific World*.
- Shaban, A. A., Dela Fuente, J. A., Salih, M. S., & Ali, R. I. (2023). Review of swarm intelligence for solving symmetric traveling salesman problem. *Qubahan Academic Journal*, 3(2), 10–27.
- Marqas, R. B., Almufti, S. M., Othman, P. S., & Abdulrahman, C. M. (2020). Evaluation of EHO, U-TACO and TS metaheuristics algorithms in solving TSP. *Journal of Xi'an University of Architecture & Technology*, 12(4), 3245–3246.
- Almufti, S. M. (2022b). Lion algorithm: Overview, modifications and applications. *International Research Journal of Science, Technology, Education, and Management*, 2(2), 176

Chapter 12: Optimization Problem: Models, Methods, and Benchmark Analysis

1. Introduction

Optimization lies at the core of modern engineering and operations research, providing systematic methods for improving performance, reducing costs, and maximizing efficiency across a wide range of industries. From the routing of delivery vehicles and the scheduling of manufacturing systems to the structural design of mechanical components, optimization problems are ubiquitous. These problems often involve multiple objectives, nonlinear constraints, and complex interdependencies, making them challenging to solve using classical methods alone.

While early optimization efforts relied on exact analytical methods such as linear programming, integer programming, and dynamic programming, the computational burden and problem-specific limitations of these approaches have led to a shift toward more flexible, general-purpose algorithms (Almufti, Shaban, Ali, Ali, & Dela Fuente, 2023). In recent decades, metaheuristic algorithms—such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Grey Wolf Optimizer (GWO)—have emerged as powerful alternatives for solving complex, real-world optimization problems that are otherwise intractable. These methods are particularly well-suited to handling multimodal search spaces, large-scale combinatorial problems, and constrained nonlinear systems (Almufti, 2019).

Benchmark problems serve a critical role in evaluating and comparing optimization algorithms. They provide standardized formulations with known properties (e.g., global optima, modality, constraints) and allow researchers to test convergence speed, robustness, accuracy, and constraint satisfaction under controlled conditions. Commonly used mathematical benchmarks include the Sphere, Rosenbrock, Rastrigin, and Ackley functions. Additionally, engineering design benchmarks such as the Pressure Vessel Design, Tension/Compression Spring Design, and the Traveling Salesman Problem (TSP) provide realistic, application-driven test beds that mimic the challenges faced in industrial settings.

Despite the rapid evolution of optimization techniques, the literature remains fragmented across algorithmic domains, problem types, and benchmarking practices.

2. Literature Review

Over the past three decades, metaheuristic algorithms have evolved into essential tools for addressing complex optimization problems that are beyond the capabilities of traditional deterministic methods. These algorithms—typically inspired by natural, biological, or social phenomena—are particularly effective in navigating the vast, multimodal, and constraint-laden landscapes encountered in engineering and combinatorial optimization.

The Ant Colony Optimization (ACO) algorithm, introduced by Dorigo and Stützle (2004), has been widely adopted for discrete optimization problems, especially the Traveling Salesman Problem (TSP) (Bonabeau et al., 1999), due to its pheromone-based learning mechanism. However, its application to continuous domains such as Welded Beam Design (WBD) and Pressure Vessel Design (PVD) often necessitates transformation through discretization or hybridization (Almufti, 2017).

Lion Algorithm (LA), although relatively recent, models the social dynamics of lion prides for optimization, offering promising results in structural and mechanical engineering problems (Mehrabian & Lucas, 2010). Its adaptability to both continuous and constrained search spaces positions it well for problems like PVD and TSD.

Cuckoo Search (CS), proposed by Yang and Deb (2009), leverages Lévy flights to enable global search and has demonstrated robust performance across both discrete and continuous domains. In particular, its application to structural design and resource allocation problems has gained traction due to its low parameter sensitivity and strong exploration capability (Almufti et al., 2025).

Grey Wolf Optimizer (GWO), introduced by Mirjalili et al. (2014), mimics the leadership hierarchy and hunting strategy of grey wolves. GWO is especially effective in real-valued constrained optimization tasks, as demonstrated in applications to WBD and TSD (Marqas et al., 2021). Its simplicity and strong exploitation mechanism make it one of the most popular recent swarm algorithms.

The Vibrating Particles System (VPS), a physics-inspired optimizer, models particle dynamics influenced by local and global attractors. It has been effectively used for continuous design problems like the Tension Spring Design, particularly due to its multi-directional update formula that balances convergence and diversity (Siddique & Adeli, 2018).

Social Spider Optimization (SSO), developed by Cuevas et al. (2014), incorporates gender-based roles and social communication among spiders. Its unique vibration-based operator enables adaptability in constrained engineering problems, though its discrete version is less mature compared to ACO or ABC.

Cat Swarm Optimization (CSO) combines seeking (exploration) and tracing (exploitation) modes, drawing inspiration from feline behavior. Studies have validated its use in biomedical engineering, structural optimization, and binary feature selection, though its sensitivity to parameter tuning remains a challenge (Chu et al., 2009).

The Bat Algorithm (BA), formulated by Yang (2010), employs frequency modulation and echolocation principles to adjust velocity and position updates. It has been successfully applied to various engineering design tasks (e.g., WBD, PVD) due to its high convergence speed and dynamic adjustment strategies (Al-Attar & Almufti, 2022).

Finally, the Artificial Bee Colony (ABC) algorithm, introduced by Karaboga (2005), simulates the food foraging behavior of honey bees. ABC has been extensively used for constrained problems like KP and PVD, where its division into employed, onlooker, and scout bees ensures a balance between local exploitation and global exploration (Almufti et al., 2022).

While several comparative studies exist (Deb et al., 2002), few provide a unified framework evaluating these algorithms across both combinatorial and continuous benchmarks. The current chapter fills this gap by categorizing problems, formalizing mathematical formulations, and aligning algorithmic suitability with domain-specific requirements.

3. Optimization Problems Categories

Optimization problems are mathematical formulations that seek to identify the best solution from a set of feasible alternatives, subject to a defined objective function. These problems are classified into distinct categories based on various criteria such as the nature of the decision variables, the structure of the objective function, the presence of constraints, and the dynamics of the problem environment table 12-1. The key categories are described as follows:

3.1. Combinatorial (Discrete) Optimization Problems

Combinatorial optimization involves problems where the decision variables are discrete and often finite, such as integers or binary values. These problems are prevalent in domains where selections, permutations, or assignments are required. For example, the Traveling Salesman Problem (TSP) seeks the shortest possible route that visits a set of cities exactly once and returns to the origin, while the Knapsack Problem (KP) requires the selection of items with maximum profit under a weight constraint. These problems are NP-hard, meaning that the computational time grows exponentially with problem size, making them ideal candidates for metaheuristic approaches like ACO, GA, and CS (Almufti, 2019).

3.2. Continuous Optimization Problems

In contrast to discrete optimization, continuous problems involve real-valued variables, where the search space is infinite within a given range. These problems frequently arise in engineering design and control systems. Examples include the Welded Beam Design (WBD), Pressure Vessel Design (PVD), and Tension/Compression Spring Design (TSD). The objective functions in these problems are often nonlinear and involve multiple real-valued parameters and nonlinear constraints. Solving such problems requires metaheuristics capable of precise exploitation and sensitive gradient-free exploration, such as GWO, BA, and VPS (Almufti, 2022).

3.3. Constrained Optimization Problems

Many real-world problems impose restrictions on the solution space, such as physical limitations, resource bounds, or performance criteria. These constraints can be:

- Equality constraints (e.g., $h(x)=0$)
- Inequality constraints (e.g., $g(x)\leq 0$)
- Bound constraints (e.g., $x_i^{(min)} \leq x_i \leq x_i^{(max)}$)

Problems such as PVD and WBD fall into this category. Effective constraint-handling strategies are essential, including penalty functions, repair mechanisms, feasibility rules, or hybridization with deterministic solvers. Algorithms like ABC and CSO often incorporate such strategies for improved feasibility search.

3.4. Unconstrained Optimization Problems

These are simpler optimization tasks where the goal is to optimize the objective function without any explicit constraints. While less common in real-world scenarios, unconstrained problems serve as essential test functions for evaluating algorithmic behavior, such as convergence speed, robustness, and precision. Standard benchmark functions like Sphere, Rosenbrock, and Rastrigin fall into this class.

3.5. Single-objective vs. Multi-objective Optimization

- **Single-objective optimization** aims to minimize or maximize a single scalar objective function. Most classical optimization problems fall into this category.
- **Multi-objective optimization (MOO)** involves optimizing two or more conflicting objectives simultaneously. For instance, in structural design, one may seek to minimize weight while maximizing strength. MOO problems do not have a single optimal solution but rather a set of Pareto-optimal solutions representing trade-offs. Advanced algorithms such as NSGA-II, MOPSO, or MOABC are specifically designed for MOO tasks.

3.6. Static vs. Dynamic Optimization Problems

- **Static optimization problems** have fixed parameters, constraints, and objective functions throughout the search process.
- **Dynamic optimization problems**, on the other hand, involve environments where one or more components (e.g., constraints, objective coefficients) change over time. Such problems are prevalent in adaptive systems, robotics, and online scheduling. Algorithms used here require real-time adaptation, memory mechanisms, and environmental prediction, as seen in dynamic variants of PSO or evolutionary algorithms.

3.7. Deterministic vs. Stochastic Optimization Problems

- **Deterministic problems** yield the same output for a given input consistently. These are well-behaved in terms of reproducibility and often allow for analytical modeling.
- **Stochastic problems** incorporate randomness either in the problem formulation (e.g., probabilistic constraints) or evaluation (e.g., noisy objective functions, Monte Carlo simulations). These are common in uncertain environments like financial modeling, risk analysis, and real-time control systems. Solving stochastic problems requires robust, noise-tolerant algorithms such as stochastic variants of GA or ABC.

3.8. Single-solution vs. Population-based Optimization

- **Single-solution methods** (e.g., Simulated Annealing) focus on iteratively improving a single candidate.
- **Population-based methods**, such as most swarm intelligence and evolutionary algorithms, maintain a set of solutions simultaneously, enabling better exploration and diversity management. These are particularly effective in multimodal and large-scale problem spaces (Almufti, Marqas, Othman, & Sallow, 2021).

Table 12-1: optimization problem by categories

Optimization Category	Example Problems/Applications
Combinatorial (Discrete)	TSP, KP, Scheduling
Continuous	WBD, PVD, TSD
Constrained	Design with limits, Resource allocation
Unconstrained	Benchmark functions (Sphere, Rosenbrock)
Single-objective	Minimize cost, Maximize efficiency
Multi-objective	Weight vs. Strength in design
Static	Fixed structure problems
Dynamic	Online scheduling, Adaptive routing
Deterministic	Linear programming, Static systems
Stochastic	Monte Carlo, Financial modelling
Single-solution	Real-time path planning, Robot motion control
Population-based	Structural design, Feature selection, Portfolio optimization

4. Mathematical Formulations of Optimization Problems

Optimization problems can be mathematically expressed as the process of minimizing or maximizing an objective function subject to a set of constraints. These problems can be classified into various types depending on the nature of the variables, the complexity of the objective landscape, and the presence or absence of constraints. In this section, we present a general formulation followed by a selection of benchmark problems that are widely used in both theoretical studies and practical engineering applications.

4.1. General Formulation

A general nonlinear constrained optimization problem is formulated as:

$$\begin{aligned}
 & \min_{x \in \mathbb{R}^n} && f(x) \\
 & \text{subject to} && g_i(x) \leq 0, i = 1, \dots, m \\
 & && h_j(x) = 0, j = 1, \dots, p \\
 & && x_i^{(min)} \leq x_i \leq x_i^{(max)}, i = 1, \dots, n
 \end{aligned}$$

Where:

$f(x)$ is the objective function to be minimized,

$x \in \mathbb{R}^n$ is the decision variable vector,

$g_i(x)$ are inequality constraints,

$h_j(x)$ are equality constraints,

x_i^{min}, x_i^{max} are variable bounds.

4.2. Travelling Salesman Problem (TSP)

Travelling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization (Laporte, 1992; Johnson & McGeoch, 2002), which has a huge search space that it cannot be solve easily (Garey & Johnson, 1979; Louis & Gong, 2000). Given a set of cities $\{C_1, C_2, C_3, \dots, C_n\}$ in which every city must be visited once only and return to the starting city for completing a tour such that the length of the tour is the shortest among all possible tours. For each pair of cities $\{C_i, C_j\}$ the distance of arc connecting those cities denoted by $d(C_i, C_j)$, the best tour for a salesman specified by an order permutation (π) of cities that have minimum tour length.

Generally there are two different kinds of TSP problems, Symmetric TSP (STSP) and Asymmetric TSP (ATSP). For the STSP the distance $d(C_i, C_j) = d(C_j, C_i)$, for n cities the number of possible tours is $(n-1)!/2$, whereas for ATSP, where the distance $d(C_i, C_j) \neq d(C_j, C_i)$, for n cities the number of possible tours is $(n-1)!$, for large number (n) of cities it is very difficult to find the exact best tours in both ATSP $(n-1)!$ and STSP $(n-1)!/2$, that is why TSP considers one of the NP-hard problems (Johnson & Papadimitriou, 1985).

Formally, the TSP is a complete weighted graph $G(N, A)$ where N is the set of cities which must be visited, and $A(i, j)$ is the set of arcs connecting the city (i) and city (j). The length between city A_i and A_j can be represented as d_{ij} (Dorigo, 2004). Thus the tour length to the given TSP problems can be found by finding the summation of the length between the cities of a permutation list as shown in formula (1).

$$Tourlength = \left(\sum_{i=1}^{n-1} d_{\pi(i) \pi(i+1)} \right) + d_{\pi(n) \pi(1)}$$

Where π is the permutation list of cities.

4.3. Benchmark Problem Formulations

Benchmark problem formulations serve as the cornerstone of algorithm evaluation in optimization research. They provide structured, replicable environments that allow researchers to systematically assess the performance, generalization ability, and robustness of optimization algorithms across diverse scenarios. These problems span both mathematical test functions—designed to evaluate specific algorithmic characteristics such as convergence behaviour, sensitivity to initial conditions, and ability to escape local optima—and practical engineering design problems that reflect the complexity, constraints, and trade-offs encountered in real-world applications. By encompassing continuous, discrete, and mixed-variable problems, benchmark formulations facilitate fair comparison and guide the selection or development of methods tailored to problem-specific demands. In what follows, we present a curated set of ten benchmark problems—including six classical test functions and four widely

studied engineering applications—detailing their mathematical structures, dimensionality, and optimization objectives see table 12-2.

Table 12-2: 1-10 benchmark problem

Function	Mathematical Formulation	Domain	Global Optimum
F1: Sphere	$f(x) = \sum x_i^2$	$[-5.12, 5.12]^D$	$f(0, \dots, 0) = 0$
F2: High Conditioned Elliptic	$f(x) = \sum (10^6)^{\left\{\frac{i-1}{D-1}\right\}} x_i^2$	$[-5, 10]^D$	$f(0, \dots, 0) = 0$
F3: Discus	$f(x) = 10^6 x^{12} + \sum_{i=2 \text{ to } D} x_i^2 (i$	$[-10, 10]^D$	$f(0, \dots, 0) = 0$
F4: Rosenbrock	$f(x) = \sum [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-10, 10]^D$	$f(1, \dots, 1) = 0$
F5: Ackley	$f(x) = -20 \exp\left(-0.2 \frac{1}{D} \sum x_i^2\right) - \exp\left(\frac{1}{D} \sum \cos(2\pi x_i)\right) + 20 + e$	$[-5.12, 5.12]^D$	$f(0, \dots, 0) = 0$
F6: Weierstrass	$f(x) = \sum \sum 0.5^k \cos(2\pi 3^k (x_i + 0.5)) - D \sum 0.5^k \cos(2\pi 3^k \cdot 0.5)$	$[-0.5, 0.5]^D$	≈ 0 after bias correction
F7: Schaffer's F7	$f(x) = \left[\frac{1}{D-1} \sum ((x_i^2 + x_i^{+12})^{0.25} (\sin^2(50(x_i^2 + x_i^{+12})^{0.1}) + 1)) \right]^2$	$[-100, 100]^D$	$f(0, \dots, 0) = 0$
F8: Griewank	$f(x) = \frac{1}{4000} \sum x_i^2 - \prod \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	$f(0, \dots, 0) = 0$
F9: Rastrigin	$f(x) = \sum [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	$f(0, \dots, 0) = 0$
F10: Shifted Rotated Griewank	$f(x) = 1/4000 \sum y_i^2 - \prod \cos(y_i/\sqrt{i}) + 1, y = M(x - o)$	$[-600, 600]^D$	$f(o) = 0$

4.4. Knapsack Problem (KP)

The Knapsack Problem (KP) is one of the most well-known NP-complete combinatorial optimization problems, extensively studied in operations research, computer science, and applied optimization. Its structure models resource allocation under capacity constraints, making it foundational for logistics, finance, telecommunications, and embedded systems applications. Due to its simplicity in formulation and complexity in solution, KP is frequently used as a benchmark for testing metaheuristic and exact algorithms.

Given:

- A set of n items,
- Each item i has a value $v_i \in \mathbb{R}^+$ and a weight $w_i \in \mathbb{R}^+$,
- A knapsack with maximum weight capacity W ,

The objective is to:

$$\begin{aligned} & \max \sum v_i x_i \\ & \text{subject to } \sum w_i x_i \leq W \end{aligned}$$

where $x_i \in \{0,1\}$ indicates whether item i is included.

4.5. Other Problem Formulations

A. Optimizing Energy Efficiency in Wireless Sensor Networks

In wireless sensor networks (WSNs), energy efficiency is critical to prolonging the lifetime of the network. The objective function can be defined as:

$$f(x) = \sum E_i(x) + \alpha \cdot \sum D_i(x),$$

where:

- $E_i(x)$: Energy consumption of the i -th sensor,
- $D_i(x)$: Distance of the i -th sensor to its target,
- α : Weighting factor.

B. Feature Selection in Machine Learning

Feature selection aims to reduce dataset dimensionality while maintaining high classification accuracy. The objective function can be defined as:

$$f(x) = w^1 \cdot (1 - ACC(x)) + w^2 \cdot |x|,$$

where:

- $ACC(x)$: Classification accuracy of the selected features x ,
- $|x|$: Number of selected features,
- w_1, w_2 : Weighting factors balancing accuracy and feature count.

C. Optimizing Power Flow in Smart Grids

In smart grids, optimizing power flow reduces energy losses and ensures load balancing. The objective function is often formulated as:

$$f(x) = \sum P_i(x) + \sum L_i(x),$$

where:

- $P_i(x)$: Power supplied to the i -th node,
- $L_i(x)$: Energy loss in the transmission line to the i -th node.

D. Structural Optimization in Engineering

Structural optimization involves minimizing material usage while ensuring

mechanical stability. For a truss structure, the objective function can be expressed as:

$$f(x) = \Sigma(W_i(x)) + \beta \cdot \Sigma(S_i(x)),$$

where:

- $W_i(x)$: Weight of the i-th truss element,
- $S_i(x)$: Stress in the i-th truss element,
- β : Weighting factor for stress constraints.

E. tension/compression spring design problem

Oscillations around an equilibrium state are caused by vibration, which is a mechanical phenomenon. it is a well-known problem that belongs to Single-Objective Constrained Optimization Problems (SOCOP) within the engineering discipline. There are two types of vibration: (1) free vibration and (2) forced vibration

An example of the continuous constrained problem, shown in fig. 12-1. is the tension/compression spring design problem (TCSD). The goal is to reduce the coil spring's volume under a constant tension/compression load. This problem is made up of three design variables. Which are:

$$P = x_1 \in [2, 15]$$

$$D = x_2 \in [0.25, 1.3]$$

$$d = x_3 \in [0.05, 2].$$

Where P is the number of spring's active coils, D is the winding diameter, and d represents the wire diameter. These parameters are used to lower weight while abiding by restrictions on flow frequencies, sheared stress, and minimum deflections.

The TCSD problem can be mathematically written using the cost function Eq(1), which must be minimized with constraints g_1, g_2, g_3 and g_4 in Eq(2):

$$f(x) = (X_3 + 2)X_2X_1^2 \quad (1)$$

The space of the design is constrained by X_1, X_2 , and X_3 values. The range of design-variables is constrained by the Lower-Boundary (Lb) = $[0.05, 0.25, 2]$ and the Upper-Boundary (Ub) = $[2, 1.3, 15]$. Relying on the descriptions for the modelling of a COP and penalty received in Eq (1). Four constraints are concerning the shear stress, surge frequency, and the minimum deflection. One objective function, three design variables, and all four constraints can be seen in Eq(2)

$$\begin{aligned} g_1(x) &= 1 - \frac{X_2^3 X_3}{72785 X_1^2} \leq 0 \\ g_2(x) &= \frac{4X_2^2 - X_1 X_2}{12566(X_2 X_1^3) - X_1^4} + \frac{1}{5108 X_1^2} - 1 \leq 0 \end{aligned} \quad (2)$$

$$g_3(x) = 1 - \frac{140.45X_1}{X_2^2 X_3} \leq 0$$

$$g_4(x) = \frac{X_1 + X_2}{1.5} - 1 \leq 0$$

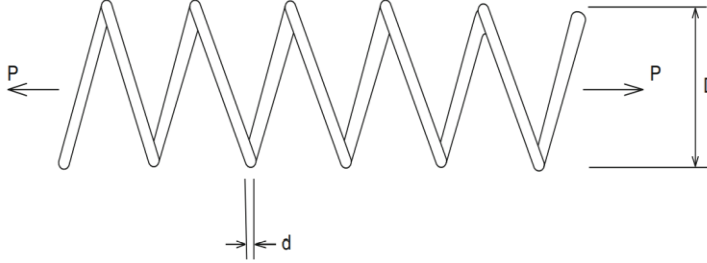


Fig. 12-1: Schematic of the tension/compression spring

In an attempt to test the effectiveness of recently developed artificial bee colony metaheuristic algorithm. The tension/compression spring design problem is selected. Which is a well-known engineering optimization problem.

F. Pressure Vessel Design (PVD) Problem

Fig. 12-2 depicts a pressure vessel design model with four choice variables: The total variables are (x_1, x_2, x_3, x_4) , where x_1 is the thickness of the pressure vessel T_s , x_2 is the thickness of the head T_h , x_3 is the inner radius of the vessel R , and x_4 is the length of the vessel barring head L . Minimizing the overall cost, which includes the cost of the material, the cost of forming, and the cost of welding, is the problem's objective function. Consequently, the following is how the basic pressure vessel design optimization model is expressed as follows [8]:

Find:

$X = (X_1, X_2, X_3)$ for (T_s, T_h, R, L)

To minimize

$$\min f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \quad (1)$$

Subject to:

1-Hoop stress $\leq$ Allowable stress

$$g_1(x) = -x_1 + 0.0193x_3 \leq 0 \quad (2)$$

2- Longitudinal stress $\leq$ Allowable stress

$$g_2(x) = -x_2 + 0.00954x_3 \leq 0 \quad (3)$$

3- Volume $\leq$ Desired volume

$$g_3(x) = -\pi x_3^2 x_4 - \frac{4}{3} \pi x_3^2 + 1296000 \leq 0 \quad (4)$$

4- Length of Shell

$$g_4(x) = x_4 - 240 \leq 0 \quad (5)$$

where $1*0.0625 \leq x_1 \leq 99$, $1*0.0625 \leq x_2 \leq 99$, $10 \leq x_3 \leq 240$, $10 \leq x_4 \leq 240$.

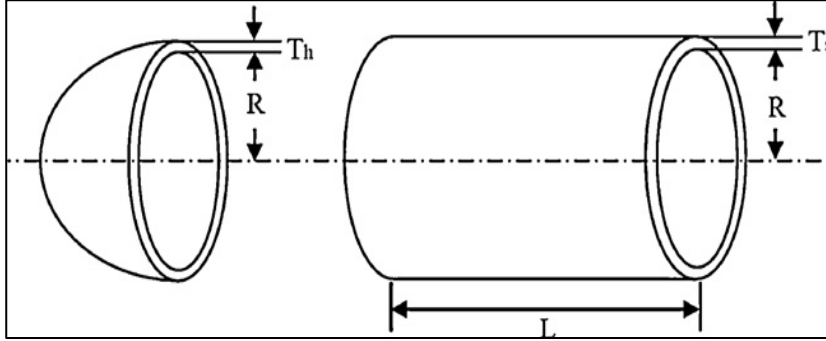


Fig. 12-2: Pressure Vessel Design

G. Welded Beam Design (WBD)

Welded Beam Design that is shown in fig. 12-3, is an engineering Single-Objectives Constrained Optimizations Benchmark Problems. It involves designing a welded beam with the lowest possible cost while taking into account side limitations, shear stress(τ), bending stress, buckling (σ), load on the bar (P_c), and end deflection (δ). Four variables make up the design: h (x_1), l (x_2), t (x_3), and b (x_4). This issue may be expressed quantitatively as follows:

$$\min f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \quad (1)$$

$$\text{s. t. } g_1(x) = \tau(x) - \tau_{\max} \leq 0 \quad (2)$$

$$g_2(x) = \sigma(x) - \sigma_{\max} \leq 0 \quad (3)$$

$$g_3(x) = x_1 - x_4 \leq 0 \quad (4)$$

$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0 \quad (5)$$

$$g_5(x) = 0.125 - x_1 \leq 0 \quad (6)$$

$$g_6(x) = \delta(x) - \delta_{\max} \leq 0 \quad (7)$$

$$g_6(x) = P - P_c(x) \leq 0 \quad (8)$$

$$\text{Where } \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \quad (9)$$

$$\tau' = \frac{P}{2^{0.5}x_1x_2} \quad (10)$$

$$\tau'' = \frac{MR}{J} \quad (11)$$

$$M = P \left(L + \frac{x_2}{2} \right) \quad (12)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2} \right)^2} \quad (13)$$

$$J = 2 \left\{ 2^{0.5} x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2} \right) \left(\frac{x_1 + x_3}{2} \right) \right] \right\} \quad (14)$$

$$\sigma(x) = \frac{6PL}{x_4 x_3^2} \quad (15)$$

$$\delta(x) = \frac{4PL^3}{Ex_3^3 x_4} \quad (16)$$

$$P_c(x) = \frac{4.013E \sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2} \left(1 - \frac{x_3}{2L} \sqrt{\frac{E}{4G}} \right) \quad (17)$$

Where $P = 6000\text{lb}$, $L = 14\text{ in}$, $E = 30 \times 10^6\text{ psi}$, $G = 12 \times 10^6\text{ psi}$, $\tau_{\max} = 13,600\text{ psi}$, $\sigma_{\max} = 30,000\text{ psi}$, $\delta_{\max} = 0.25\text{ in}$, $0.1 \leq x_1 \leq 2$, $0.1 \leq x_2 \leq 10$, $0.1 \leq x_3 \leq 10$, $0.1 \leq x_4 \leq 2$.

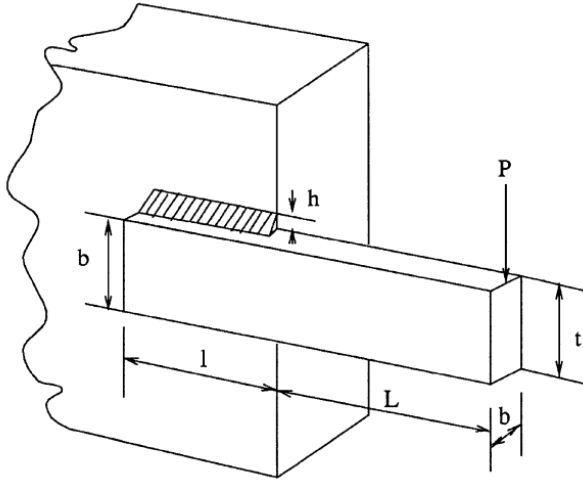


Fig. 12-3: Welded Beam Design

5. Metaheuristics algorithm

In the realm of computational optimization, metaheuristic algorithms have proven indispensable for solving a wide spectrum of engineering and combinatorial problems. These algorithms—drawing inspiration from biological, physical, or social systems—are particularly valued for their flexibility, global search capabilities, and problem-independent structures. However, their effectiveness varies significantly depending on the nature of the optimization task, such as whether it is discrete or continuous, constrained or unconstrained,

or single- or multi-objective. This section provides a comparative assessment of several prominent metaheuristic algorithms—Ant Colony Optimization (ACO), Lion Algorithm (LA), Cuckoo Search (CS), Grey Wolf Optimizer (GWO), Vibrating Particles System (VPS), Social Spider Optimization (SSO), Cat Swarm Optimization (CSO), Bat Algorithm (BA), and Artificial Bee Colony (ABC)—when applied to five benchmark problems: the Traveling Salesman Problem (TSP), Welded Beam Design (WBD), Pressure Vessel Design (PVD), Tension/Compression Spring Design (TSD), and the Knapsack Problem (KP). The comparison highlights the domain-specific strengths, adaptation requirements, and practical performance of each algorithm, offering insight into their suitability for various classes of optimization challenges.

Table 12-3: comparative analysis of metaheuristic algorithms

Algorithm	TSP	WBD	PVD	TSD	KP
ACO	✔ Excellent (uses path probability & pheromones)	⚠ Less efficient (continuous form hard)	⚠ Requires discretization	⚠ Weak in real-valued search	✔ Strong for binary KP
LA	⚠ Moderate (requires discretization)	✔ Capable (through real-encoded vectors)	✔ Good	✔ Good	✔ Suitable
CS	✔ Good (with permutation encoding)	✔ Good	✔ Very Good	✔ Efficient	✔ Good
GWO	⚠ Weak (not discrete)	✔ Excellent (good for real values)	✔ Excellent	✔ Excellent	⚠ Needs modification
VPS	⚠ Moderate (adaptable)	✔ Strong	✔ Strong	✔ Strong	✔ Good
SSO	⚠ Not ideal (discrete version complex)	✔ Good	✔ Good	✔ Good	✔ Moderate
CSO	⚠ Requires discrete adaptation	✔ Good	✔ Strong	✔ Strong	✔ Good
BA	⚠ Needs tuning for discrete	✔ Excellent	✔ Excellent	✔ Excellent	✔ Moderate
ABC	✔ Strong for combinatorial KP	✔ Good for constrained design	✔ Good	✔ Moderate	✔ Excellent

Table 12-3 shows the comparative analysis of metaheuristic algorithms across key optimization problems reveals diverse strengths and limitations. Ant Colony Optimization (ACO) excels in solving the Traveling Salesman Problem (TSP) due to its path probability modeling and pheromone-based learning, and performs strongly on the Knapsack Problem (KP), yet struggles with real-valued problems like Welded Beam Design (WBD), Pressure Vessel Design (PVD), and Tension/Compression Spring Design (TSD) due to the need for discretization. The Lion Algorithm (LA), though moderate on TSP, proves competent across

WBD, PVD, TSD, and KP by leveraging real-encoded vectors. Cuckoo Search (CS) demonstrates balanced performance across all problems with good to very good adaptability in both discrete and continuous domains. Grey Wolf Optimizer (GWO) is highly effective for real-valued problems (WBD, PVD, TSD), though its performance in discrete tasks like TSP and KP requires structural adjustments. The Vibrating Particles System (VPS) shows strong performance in real-valued engineering problems and moderate adaptability in combinatorial tasks. Social Spider Optimization (SSO) and Cat Swarm Optimization (CSO) require discrete adaptation for problems like TSP, yet both offer solid performance in structural and real-valued problems, with SSO being moderate on KP. The Bat Algorithm (BA) is highly effective for WBD, PVD, and TSD, but needs parameter tuning for discrete problems. Finally, Artificial Bee Colony (ABC) stands out in KP and offers competent solutions for engineering design problems, though its performance in TSD is comparatively moderate.

6. Discussion

The diversity of optimization problems in engineering and computer science necessitates a careful match between problem structure and algorithmic strategy. This study demonstrates that no single metaheuristic excels universally; rather, each algorithm's performance is problem-dependent. For instance, ACO and ABC showcase superior capabilities for discrete combinatorial tasks like the TSP and KP, leveraging probabilistic encoding and adaptive search. Conversely, GWO, BA, and VPS perform robustly on real-valued, continuous problems such as WBD, PVD, and TSD, due to their exploitation mechanisms and natural compatibility with continuous domains.

The CS algorithm offers a balance between exploration and exploitation, delivering stable results across both problem types, while LA, SSO, and CSO require careful tuning or structural modifications, particularly for discrete spaces. A striking observation is that constrained optimization scenarios, such as WBD and PVD, benefit greatly from algorithms with embedded constraint-handling strategies—most notably ABC and CSO.

Moreover, the classification of problems into categories such as single vs. multi-objective, static vs. dynamic, and deterministic vs. stochastic provides a conceptual framework for future algorithm selection. For example, real-time adaptive systems may prefer population-based algorithms like GWO or CSO for their inherent diversity, whereas deterministic design tasks could rely on the precision of BA or VPS.

This analysis underscores the importance of benchmarking in developing robust optimization techniques. By anchoring algorithm evaluation in standardized problems—both mathematical and application-driven—researchers can better diagnose algorithmic behaviours and drive innovation in hybrid, adaptive, or problem-specific algorithmic designs.

7. References

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. Oxford University Press.

- Almufti, S. M., Shaban, A. A., Ali, Z. A., Ali, R. I., & Dela Fuente, J. A. (2023). Overview of metaheuristic algorithms. *Polaris Global Journal of Scholarly Research and Trends*, 2(2), 10–32.
- Almufti, S. M. (2019). Historical survey on metaheuristics algorithms. *International Journal of Scientific World*, 7(1), 1–12. <https://doi.org/10.14419/ijsw.v7i1.29497>
- Chu, H., Roddick, J. F., & Pan, J.-S. (2009). Cat swarm optimization for feature selection. *Expert Systems with Applications*, 36(3), 7014–7025. <https://doi.org/10.1016/j.eswa.2008.08.042>
- Cuevas, E., Zaldivar, D., & Pérez-Cisneros, M. (2014). Social spider optimization. *Applied Soft Computing*, 24, 303–318. <https://doi.org/10.1016/j.asoc.2014.07.011>
- Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197. <https://doi.org/10.1109/4235.996017>
- Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. MIT Press.
- Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization (Technical Report-TR06). Erciyes University.
- Marqas, R. B., Almufti, S. M., Ahmed, H. B., & Asaad, R. R. (2021). Grey wolf optimizer: Overview, modifications and applications. *International Research Journal of Science, Technology, Education, and Management*, 1(1), 44–56.
- Mehrabian, A. R., & Lucas, C. (2010). A novel numerical optimization algorithm inspired from the behavior of lions. *Scientia Iranica*, 17, 424–433.
- Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69, 46–61. <https://doi.org/10.1016/j.advengsoft.2013.12.007>
- Siddique, B., & Adeli, H. (2018). Vibrating particle system algorithm for optimization. *Journal of Computing in Civil Engineering*, 32(1), 04017080. [https://doi.org/10.1061/\(ASCE\)CP.1943-5487.0000715](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000715)
- Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In J. R. González et al. (Eds.), *Nature inspired cooperative strategies for optimization (NICSO 2010)* (pp. 65–74). Springer. https://doi.org/10.1007/978-3-642-12538-6_6
- Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. In *Proceedings of the World Congress on Nature & Biologically Inspired Computing* (pp. 210–214). <https://doi.org/10.1109/NaBIC.2009.5393690>
- Al-Attar, M. R., & Almufti, S. M. (2022). Bat algorithm performance in structural engineering optimization. *Academic Journal of Nawroz University*, 10(3).
- Almufti, S. M. (2017). Using swarm intelligence for solving NP-hard problems. *Academic Journal of Nawroz University*, 6(3), 46–50. <https://doi.org/10.25007/ajnu.v6n3a78>
- Almufti, S. M., Marqas, R. B., Asaad, R. R., & Shaban, A. A. (2025). Cuckoo search algorithm: Overview, modifications, and applications. *International Journal of Scientific World*, 11(1), 1–9.

- Almufti, S. M., Alkurdi, A. A., & Khoursheed, E. A. (2022). Artificial bee colony algorithm performance in solving constraint-based optimization problems. *Temematique*, 21(1)
- Almufti, S. M., Marqas, R. B., Othman, P. S., & Sallow, A. B. (2021). Single-based and population-based metaheuristics for solving NP-hard problems. *Iraqi Journal of Science*.

Saman M. Almufti

Metaheuristics Algorithms: Overview, Applications, and Modifications

Metaheuristic algorithms have emerged as essential tools for solving complex optimization problems across disciplines such as engineering, logistics, finance, and healthcare. Their ability to efficiently explore large, nonlinear, and uncertain search spaces makes them highly effective where traditional methods often fail.

This book, *Metaheuristics Algorithms: Overview, Applications, and Modifications*, offers a structured overview of key algorithmic families—including evolutionary, swarm-based, physics-inspired, and human-based approaches—supported by theoretical foundations, classifications, and real-world applications. Emphasis is placed on recent advancements, hybridizations, and performance-enhancing modifications.

Designed for students, researchers, and practitioners, the content balances academic rigor with practical relevance, aiming to guide both implementation and innovation in metaheuristic optimization.

I am grateful to my colleagues and reviewers for their valuable input, and to Reta N. Mussa for the attractive design of the book cover. My appreciation also extends to Deep Science Publishing for enabling its open-access dissemination. I hope this work contributes meaningfully to advancing research in computational intelligence and intelligent optimization.

